Diagnosing tree-decomposable circuits *

Yousri El Fattah
Computer Science Department
University of California at Irvine
Irvine, CA 92717
(fattah@ics.uci.edu)

Rina Dechter
Computer Science Department
University of California at Irvine
Irvine, CA 92717
(dechter@ics.uci.edu)

Abstract

This paper describes a diagnosis algorithm called *structure-based abduction* (SAB) which was developed in the framework of constraint networks [12]. The algorithm exploits the structure of the constraint network and is most efficient for near-tree problem domains. By analyzing the structure of the problem domain, the performance of such algorithms can be bounded in advance. We present empirical results comparing SAB with two model-based algorithms, MBD1 and MBD2, for the task of finding one or all minimal-cardinality diagnoses. MBD1 uses the same computing strategy as algorithm GDE [9]. MBD2 adopts a breadth-first search strategy similar to the algorithm DIAGNOSE [24]. The main conclusion is that for nearly acyclic circuits, such as the N-bit adder, the performance of SAB being linear provides definite advantages as the size of the circuit increases.

1 Introduction

Generally speaking, *diagnosis* is a form of abduction or inference to the best explanation. *Explanations* are those minimal sets of value instantiations that are consistent with the model and the observations, and *best* explanations are those that optimize some desirability measure. Some common measures are *minimal cardinality* [19], *parsimonious* covering theory [23], *most probable* explanation [22], and *minimal cost proofs* [2]. A well-defined relation between cost-based abduction and belief revision in probabilistic networks has been shown [2], and also algorithms based on systems of linear constraints have been proposed [26].

Diagnosis, when viewed as cost-based abduction, can be formulated as an optimization task in the framework of constraint networks [12; 11; 18]. The advantage of such formulation is that all algorithms and heuristics developed within that framework can be exploited. In particular, it was shown that when a constraint network is acyclic, an optimal solution can be found in linear time [13]. Tailoring this algorithm to diagnosis results in an algorithm called *structure-based abduction* (SAB) [12], which will be empirically investigated here. The performance of SAB is compared with two model-based diagnosis algorithms called here MBD1 and MBD2. MBD1 uses the same strategy for finding predictions and conflicts as GDE [9]. MBD2 is a focused version of MBD1 geared toward computing minimal-cardinality diagnoses. MBD2 adopts a breadth-first search strategy for searching the hypotheses space and then computes conflicts sequentially, as in DIAGNOSE by Reiter [24].

The main contribution of this work is in illustrating how the task of diagnosis can be performed efficiently for circuits whose structure corresponds to nearly acyclic networks. Note that model-based diagnosis algorithms like GDE [9] can take an exponential time even on acyclic circuits (see Resnick [25]). We present results of experiments on a parameterized circuit that generalizes Resnick's circuit showing instances where the GDE-like algorithm (MBD1) explodes in time and space while SAB is able to compute diagnoses efficiently for all instances. We then show how SAB can be applied to circuits whose structure is cyclic by using tree clustering [14] as a preprocessing phase. Applying tree clustering to cyclic circuits is known to be exponential in the cluster's sizes. We give empirical results on a parameterized cyclic circuit showing that even if relatively expensive, it is worthwhile doing tree clustering once so that all future diagnosis tasks are performed efficiently.

The structure of the paper is as follows. Section 2 reviews basic definitions and presents two model-based diagnosis algorithms: MBD1, MBD2. Section 3 formulates diagnosis as optimization in constraint networks and describes algorithm SAB, and section 4 describes the tree clustering task. Section 5 presents empirical results, section 6 provides discussion and related work, and section 7 concludes.

2 Model-Based Diagnosis

Following [7] we define *model-based diagnosis* in terms of a triple $(SD, COMPS, OBS)$ where SD, the system description, is a set of first-order sentences; $COMPS$, the system components, is a set of constants; OBS, the

*This work was supported in part by the NSF (grant IRI-9157636), the Air Force Office of Scientific Research (grant AFOSR 900136), Toshiba of America, Xerox, Northrop and Rockwell.

observations, is a set of first-order sentences. Intuitively, a diagnosis is a subset of components $\Delta \subset COMPS$ such that assuming that each component in Δ is faulty while the components in $COMPS - \Delta$ are not faulty, is consistent with the model and the observations. A diagnosis Δ for (SD, $COMPS$, OBS) is minimal if it does not contain a proper subset diagnosis.

Two diagnosis algorithms, MBD1 and MBD2, are examined, and will be described informally due to space restrictions. Algorithm MBD1 is a variant of GDE [9]. It works as a three-step process. In step 1, predictions are computed by a form of constraint propagation, which although generally incomplete, is complete for restricted languages such as trees and Horn theories. In step 2, all minimal conflict sets (nogoods) are enumerated by identifying those "environments" leading to conflicts between predictions and observations. In step 3, all minimal-cardinality diagnoses are computed as the minimal-cardinality covers of all the conflict sets.

The worst-case time and space complexity for finding all minimal conflict sets is clearly exponential in $|COMPS|$. Since the task of finding a minimal cover to a set of subsets is known to be NP-complete (even when each subset has at most two elements), the resulting worst-case complexity of the algorithm is likely to be (if $P \neq NP$) exponential.

The second diagnosis algorithm MBD2 reverses the steps of MBD1. It first generates a hypothetical diagnosis and then determines whether any conflicts remain unexplained by that hypothetical diagnosis. If such a conflict is found then the hypothesis is extended to cover each component in the new conflict. If such a conflict does not exist, the hypothesis is a minimal diagnosis. All other minimal-cardinality diagnoses are found by checking each of the remaining hypotheses as to whether it can explain all the conflicts. MBD2 is a version of the algorithm DIAGNOSE by Reiter [24] modified for computing the minimal cardinality diagnoses.

3 Diagnosis as Optimization

Given a set of variables $X_1, ..., X_n$ each having a finite set of domain values $dom(X_1), ..., dom(X_n)$, a *constraint network* CN is a set of relations $\{r_1, ..., r_m\}$, called *constraints*, each defined on a subset of variables $S_1, ..., S_m$. A *relation* over a subset of variables $X_{i_1}, ..., X_{i_j}$ is a subset of the cartesian products of their domains. A *solution* to CN is an assignment of a value to each variable satisfying all the constraints. A constraint network can be associated with the set of all its solutions. Formally,

$$sol(CN) = \{t = (X_1 = x_1, \ldots, X_n = x_n) | \forall j,\ t_{S_j} \in r_j\},$$

when t_S stands for the projection of a tuple t on a subset of variables S. For certain applications, it is useful to define a cost function over all solutions. A simple cost function associates a cost $c(X = x)$ with each value of a variable, and the *cost of a solution* is the sum cost over all (variable, value) pairs. The optimization task is to find a solution having an optimal cost (largest or smallest).

To capture the diagnosis task, we map the triple (SD, $COMPS$, OBS) into the relational framework as follows. The system description SD will be described in terms of two sets of variables: the *system variables*, $X_1, ..., X_n$, which are the inputs and outputs of all components with their associated finite domain values, $dom(X_1), \ldots, dom(X_n)$, and the *assumption variables*, $A = \{A_1, \ldots, A_m\}$, each associated with a component $c_j \in COMPS$ and describing its functioning status. In the simplest case, these are bi-valued variables indicating whether the component is normal (value 0) or abnormal (value 1). In the more involved case, they can index different fault models. Each component (c_j) input-output behavior under normal and abnormal conditions are described by a constraint r_j. Thus, the constraint r_j is defined over the set $R_j = \{A_j\} \cup S_j$, where S_j is the set of input and output variables for component c_j. The observations OBS is described by forcing value assignments for the corresponding variables. Non-zero costs are associated with assumption variables only. An abnormal component is assigned a cost of 1 [i.e., $c(A_j = 1) = 1$], while a normal component has a zero cost [i.e., $c(A_j = 0) = 0$].

Given a model description and a set of observations, the diagnosis task is to construct an *explanation*, namely, an assumption tuple $(A_1 = a_1, \ldots, A_m = a_m)$ that can be extended to a consistent solution $(X_1 = x_1, \ldots, X_n = x_n, A_1 = a_1, \ldots, A_m = a_m)$. The cost of an explanation is the sum of the costs associated with the assumption variables. That is,

$$C(\{A_1 = a_1, \ldots, A_m = a_m\}) = \sum_{A_j \in A} c(A_j = a_j). \quad (1)$$

The task of finding a minimal-cardinality diagnosis corresponds to a minimum-cost solution since only faulty components are assigned a positive cost in the constraint network formulation.

The topology of a constraint network can be depicted by its *dual graph*. A dual graph represents each constraint by a node and associates a labeled arc with any two nodes that share variables. If the dual constraint graph is a tree (called a join tree) or can be transformed into a tree by removing redundant arcs (in linear time), then the constraint network is said to be acyclic [20]. In that case, a consistent solution or an optimal solution can be assembled in linear time.

Example 3.1 Consider the circuit in Fig. 1a, which has two AND-gates `AND-1`, `AND-2` connected to an OR-gate `OR-1`. We model this circuit using three bi-valued assumption variables, A_1, A_2, A_3, whose values, zero or one, indicate whether the respective gate `AND-1`, `AND-2`, and `OR-1`, is normal or abnormal. Fig. 1b shows the dual graph for the circuit, where a node corresponds to the

Table 1: Constraint AND-1

A1	V1	V2	V3	Cost
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	x	x	x	1

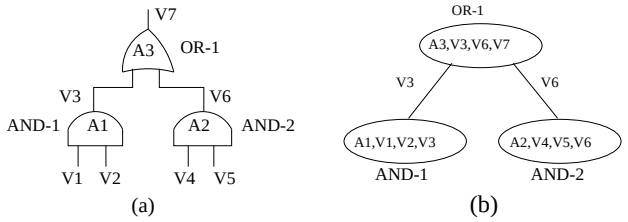

Figure 1: A circuit and its dual graph

Table 2: **SAB**

Input: A join-tree T having variables $X = \{X_1, \ldots, X_n\}$, assumption variables $A = \{A_1, \ldots, A_m\}$, a cost function c, and relations $\{r_1, \ldots, r_m\}$ defined on $R_j = \{A_j\} \cup S_j$, and $S_j \subset X$ $(j = 1, \ldots, m)$.

Output: One (or all) minimum cost solution(s).

1. (**Bottom-up**) Compute weights: If R_j is a leaf in T, then for each tuple $t \in r_j$ we define the weight mapping $w(t) = c(t_{A_j})$. If R_j is a parent in T, then for each tuple $t \in r_j$ we define the weight mapping

$$w(t) = c(t_{A_j}) + \sum_{i\,:\,child(j)} \min_{s \in r_i \mid s_{R_i \cap R_j} = t_{R_i \cap R_j}} w(s)$$

2. (**Top-down**) Generate minimum-cost solution(s) by following the pointers of a minimum-weight tuple from root to leaves.

set of input-output variables including its associated assumption variable. The explicit constraint for each node is given as a table of allowed tuples.

Constraint `AND-1` is given in Table 1, where the values marked with an `X` correspond to "don't care" conditions (either 0 or 1). Given the set of observations $\{V1 = 0, V2 = 1, V4 = 1, V5 = 1, V7 = 0\}$, there are two best explanations, both having cost 1. $\{A1 = 0, A2 = 1, A3 = 0\}$ corresponding to a single-fault of `AND-2`, and $\{A1 = 0, A2 = 0, A3 = 1\}$ corresponding to a single-fault of `OR-1`. Notice that multiple-fault explanations also exist, but they have higher costs.

Table 2 presents algorithm SAB for a join-tree network. In the bottom-up step, pointers are created from each tuple t of a parent node to each of a set of minimizing tuples for each child node. The complexity of the bottom-up phase of the algorithm is $O(n \cdot t \cdot \log t)$, where n is the number of components (nodes of the join tree) and t is the number of tuples. Details about the algorithm and arguments for correctness are given in [12; 13; 10].

4 Tree Clustering

For completeness we present in Table 3 the main steps in the tree-clustering algorithm [14]. Tree clustering is based on a triangulation algorithm which transforms any graph into a chordal graph by adding edges. The maximum cliques of the resulting chordal graph are the clusters necessary for forming an acyclic CN. The relation

Table 3: **Tree Clustering**

Input: A cyclic constraint network T.

Output: A join tree representation of T.

1. Order the nodes (in a maximum cardinality ordering (mco) and use a *triangulation algorithm* to generate a chordal graph for T.

2. Identify the maximum cliques $C_1, \ldots, C_t$ and index them by the order of their earliest variable in the ordering.

3. Return the join tree resulting from connecting each C_i to a C_j whose index $j > i$ and with whom it shares the largest set of variables.

4. Compute for each node in the tree the set of tuples satisfying all constraints on the clique and arc-consistency with the children.

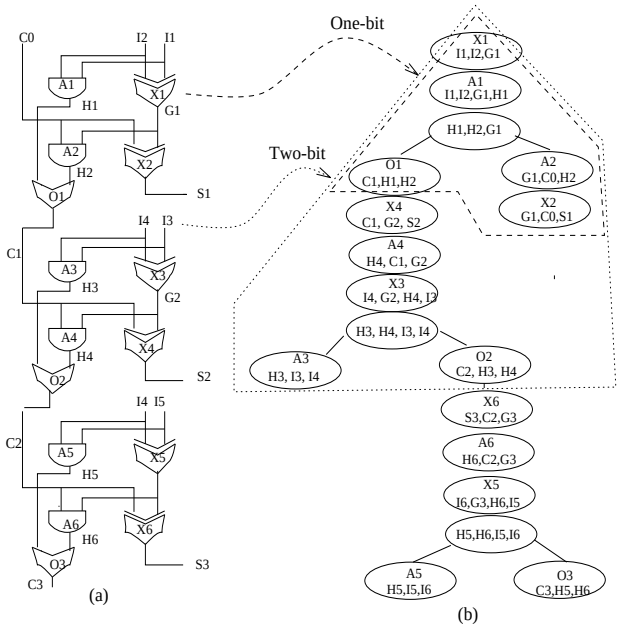

Figure 2: Tree-clustering for the three-bit adder.

for each cluster (step 3) is computed by assigning the input constraints to the various clusters of the join tree and finding for each clique all tuples satisfying those constraints. The complexity of this step dominates the algorithm's performance and is exponential in the clique's size.

Example 4.1 Figure 2 shows the join tree for the three-bit adder. We note the recursive nature of the tree, and the fact that the maximum clique size is independent of the number of bits of the adder.

5 Empirical Evaluation

5.1 Acyclic Circuits[1]

The first set of experiments is restricted to a family of circuits whose CN is acyclic. We compare the performance of SAB to MBD1 on the task of computing all

[1] Preliminary experimental results appear in [15]

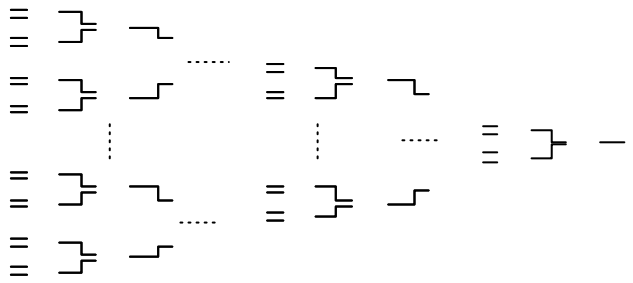

Figure 3: A parameterized circuit $b(p,k)$.

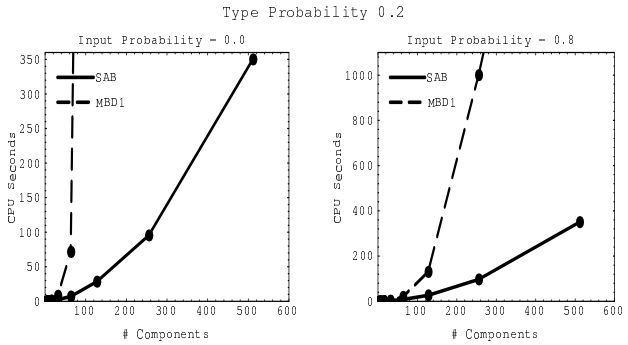

Figure 4: Average CPU seconds for different input probabilities

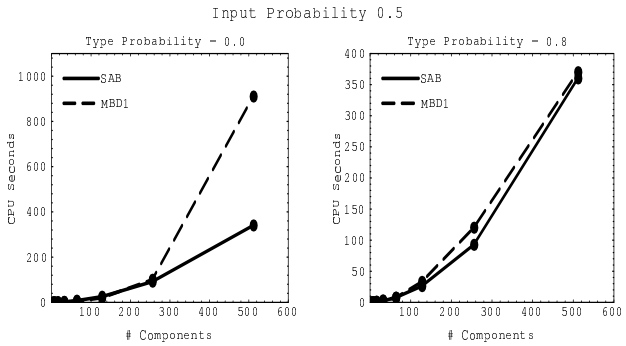

Figure 5: Average CPU seconds for different type probabilities

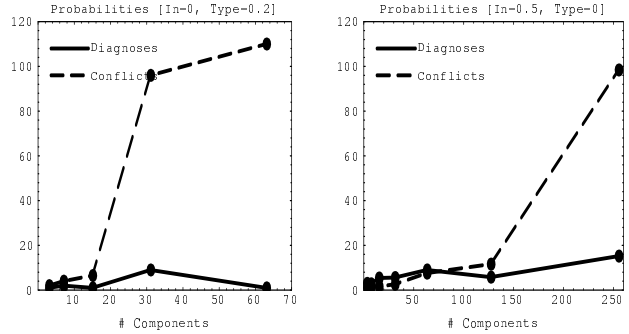

Figure 6: Average number of conflicts and diagnoses versus number of components

minimal-cardinality diagnoses on the parameterized circuit $b(p,k)$, shown in Figure 3. The circuit has $n = 2^k - 1$ components and k layers numbered $1, 2, .., k$ from the leaves to the root. The component types are independent random variables and are either AND or OR gates. A component in an odd (even) layer is an AND (OR) with probability p, OR (AND) with probability $1-p$.[2] The circuit $b(p,k)$ is acyclic and can have a worst-case exponential number of minimal conflicts.[3]

Method

For every circuit, we generate the inputs as independent random variables each having the value 1 with probability q and 0 with probability $1-q$. Then the correct output is determined and reversed. The observation consists of the inputs and the faulty output. We feed the same set of problems to both SAB and MBD1 and record their respective performances. We limit the space and time that can be used by $MBD1$ to make the experiments feasible. When the limits are reached, MBD1 is terminated with no solution. Each experiment consists of $M(=2)$ circuits each with $N(=5)$ input-output scenarios. The performance is averaged over the $N \cdot M$ problem instances. The circuits considered have components ranging from 3 to 511. Our experiments were conducted using Quintus Prolog on Sun 670MP.

[2] A special case of the circuit, $b(1,k)$, was developed by [25].

[3] Although the number of conflicts can be exponential for certain inputs, this does not imply that it is the case for most inputs.

Results

The results of the experiments, given in Figures 4 and 5, display the average CPU times used by SAB and MBD1 for computing all minimal-cardinality diagnoses. In Figure 4, the input probability is varied while the type probability (the type of the component being an AND or an OR gate) is fixed at 0.2. In Figure 5 the type probability is varied while the input probability is fixed at 0.5. Those figures show that MBD1 becomes inefficient for computing the diagnoses for moderately sized circuits. Points that are not shown for MBD1 correspond to circuit sizes for which MBD1 failed on *all* problem instances because its space (or time) requirement exceeded available resources. For such problems, the space required by MBD1's caching during the prediction phase had exhausted the available resource limit, set at 10,000 predictions. The points shown for MBD1 give the average CPU seconds over the solved-problem instances.

As observed from Figures 4 and 5, MBD1 is sensitive to the type of the randomly generated circuits and to the probability of the inputs. For circuits having 63 components and type probability 0.2, changing the input probability from 0 to 0.8 resulted in a change in the average CPU seconds from 71.5 to 19.5 for MBD1 and 6.95 to 8.04 for SAB. The coefficient of variation (the ratio of the standard deviation to the mean) of the CPU seconds is also an order of magnitude higher for MBD1 than for SAB (around 50% for MBD1 and 5% for SAB.) It is also observed that for small sized circuits, MBD1's

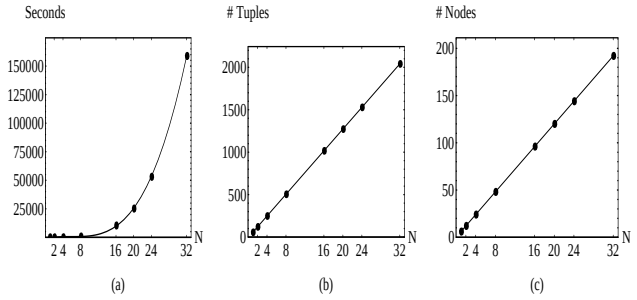

Figure 7: Performance of tree clustering for the N-bit adder.

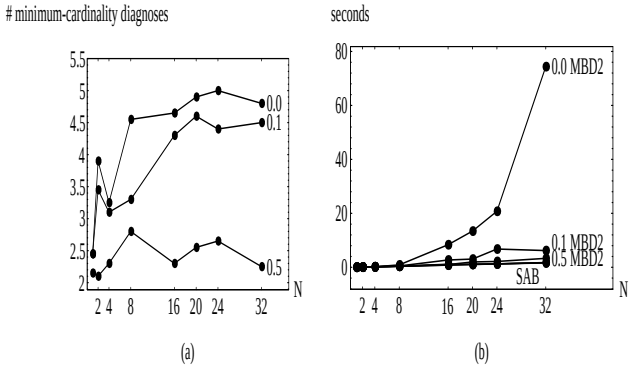

Figure 8: Performance of SAB and MBD2 for single faulty output on the N-bit adder and varying input probability.

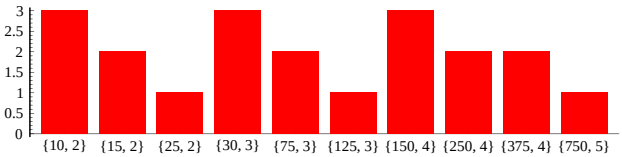

Figure 9: Histogram of number of diagnoses and their cardinality on the four-bit adder with inputs all zeros and multiple faulty outputs.

performance is comparable to SAB's. Figure 4 shows that for input probability close to 0 or 1, MBD1 is generally slower than SAB. On the other hand, for input probability 0.5, MBD1 is very close to SAB for some circuits (Figure 5). As suggested by theoretical analysis, it appears that MBD1's performance is correlated with the number of conflicts, while SAB's performance is affected by the number of diagnoses. On average, the performance of MBD1 relative to SAB degrades as the number of conflicts relative to the number of diagnoses increases. Fig. 6 shows the average number of conflicts and diagnoses for two cases: one where SAB is consistently faster (type probability 0.2 and input probability 0), and the other where MBD1 is sometimes faster (type probability 0 and input probability 0.5). The figure clearly shows that MBD1 is at a disadvantage with respect to SAB when there is a large number of conflict sets but a small number of minimal-cardinality covers. On the other hand, MBD1 is at some advantage with respect to SAB when the number of conflicts is small while the number of diagnoses is relatively large.

5.2 Cyclic Circuits

To overcome the space problem of the ATMS caching used in MBD1 in the rest of our experiments we used algorithm MBD2 since it avoids MBD1's memory problem. The algorithm is still time exponential.

In this section we demonstrate that even if the structure of the problem is not acyclic, diagnosis and abduction may still be accomplished efficiently by transforming the problem into a tree using tree clustering and subsequently solving by SAB. In this set of experiments the N-bit adder circuit, whose structure is cyclic, is considered. An N-bit adder has 5 N components and can be modeled by a network having 12 N + 1 variables.

Tree Clustering

Tree clustering is performed on the circuit prior to diagnosis. Figure 7a gives the time of tree clustering which increases polynomially with N. We also see that the total number of tuples increases linearly with N (Figure 7b), and the number of nodes of the join tree (number of cliques) grows linearly with N (Figure 7c).

Method

As in the acyclic case, we generate the inputs as independent random variables each having the value 1 with probability q and 0 with probability $1 - q$. The outputs are computed by propagating the inputs through the component's constraints. Two types of experiments are conducted. In one, a single output is randomly selected and inverted. In the second set of experiments, multiple outputs are randomly selected (half of the outputs, on average) and inverted. In all experiments, the observation consists of all input, all the correct and faulty outputs. The observation consists of all input, all correct outputs and the single faulty output. We record the performance of both SAB and MBD2 on each problem instance generated that way. An experiment consists of a set of 20 problem instances for each circuit and their performances is averaged over the problem set. A time limit of 1000 seconds per problem is set for MBD2.

Results

Figure 8 shows the performance of SAB and MBD2 for single faulty output when varying the input probabilities. A single faulty output corresponds to the case where components fail with a very small probability. The minimal-cardinality diagnoses consist of only single fault diagnoses. Figure 8a shows the average number of minimal cardinality diagnoses versus the circuit size N for three input probabilities. The average number of diagnoses is small (< 5) and decreases towards equal probabilitys for '0' and '1' (0.5). Figure 8b shows the average time for computing all minimal-cardinality diagnoses by SAB and MBD2. The figure shows that SAB is highly effective resulting in a large speed-up as N increases. The speed-up is particularly significant for the case where all inputs are zero. For input probability 0.5, the average

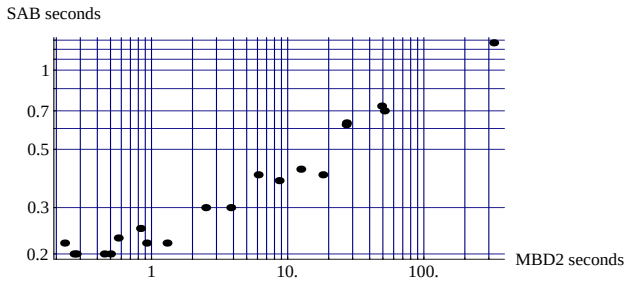

Figure 10: Time by SAB and MBD2 on the four-bit adder with inputs all zeros and multiple faulty outputs.

time is nearly identical for both SAB and MBD2. For a 16-bit adder (80 components) and inputs zeros, SAB's performance time ranges between 0.72 and 0.80 seconds, while on the same set of problems algorithm MBD2 takes between 2.3 and 11.7 seconds. For a 32-bit adder (160 components) and when all inputs are zeros, the time by SAB ranges between 1.62 and 3 sec., while by MBD2 between 30.4 and 155 sec.

For multiple faulty output, the number of minimal diagnoses increases exponentially with N and their cardinality is greater than 1. Problem instances on an 8-bit adder had 3750 minimal-cardinality diagnoses for input probability 0 and 18750 for input probability 0.1. This corresponds to the case where components fail with non-small probability. MBD2 was able to compute diagnosis up to a four-bit adder but exceeded the limit on the eight-bit adder, even when computing one diagnosis. SAB, on the other hand, computed all diagnoses for the eight-bit adder, and it took only 59.82 seconds to compute all 18750 diagnoses in one particular problem instance. For the four-bit adder, where comparison can be made between SAB and MBD2, we plotted the histogram of the distribution of 20 problem instances as a function of the number of diagnoses and their cardinality (Figure 9). We also give a scatter diagram for the time to compute all diagnoses by SAB and MBD2 on each problem instance (Figure 10). Figure 9 shows that the cardinality of the diagnoses ranged between two and five and that number of minimal-cardinality diagnoses ranged between 10 and 750. Figure 10 shows that SAB was an order of magnitude more time efficient than MBD2. MBD2's time ranged between 0.23 seconds and 325.85 seconds while SAB's time ranged between 0.2 seconds and 1.27 seconds. MBD2 required 325.85 seconds to solve a hard problem having 750 diagnoses of cardinality 5 (Figure 9) and took only 0.23 seconds to solve an easy problem having 10 diagnoses of cardinality 2 (Figure 9). SAB required 1.27 seconds for the hard problem instance, while 0.22 seconds for the easy one.

6 Discussion and related work

The common approach to model-based diagnosis has widely been based on the language of propositional logic [7] and the reasoning machinery of the ATMS [5]. The main advantage of using constraint network techniques for diagnosis is that it allows the structure of the system to be effectively exploited and the complexity of computation to be bounded in advance. For example, the circuit in Figure 3 may have an exponential number of minimal conflict sets (exponential in the square root of the number of circuit components [25]), yielding an exponential time for computing diagnoses by MBD1. Our algorithm, on the other hand, is linear time for that circuit and its efficiency can be determined by a simple analysis of the circuit structure, prior to execution. MBD2 will still perform poorly on this problem instance due to the time needed for checking that a current hypothetical diagnosis covers all conflicts. For near-tree networks, the computation done vy tree-clustering can be viewed as replacing the computation of all conflict sets in MBD1 and MBD2. However, while the number of conflicts, on which MBD1 and MBD2's performance depend, is hard to predict in advance, the cost of tree-clustering can be predicted by analyzing the structure of the circuit.

Several methods have been proposed [8; 1] to avoid the combinatorial explosion of the early GDE-style computation [9]. In those works, various heuristics are adopted that trade off completeness with efficiency. Algorithms have been proposed to compute the first k minimal (not necessarily minimal-cardinality) diagnoses (for fixed k) in polynomial time [17; 21; 3]. In [16] an approach called "structural focusing" is presented based on identifying those parts of the system whose diagnoses are independent of the remaining parts of the system. An "independent diagnosis problem" in [16] depends not only on the structure of the circuit but also on the observation instance of the diagnosis problem. A typical example is an N-bit adder problem where all inputs and outputs are set to 0 except for the output carry of the last one-bit adder. That circuit with that observation instance has first been suggested in [6] to motivate work on focusing on probable diagnosis. Unlike those focusing approaches, our approach only depends on the structure of the circuit. A structural notion of independence similar to ours has also been advocated recently and several versions of a diagnosis algorithm are given in [4].

7 Summary and conclusions

SAB is a diagnosis algorithm that exploits the structure of the problem when computing minimal-cardinality diagnoses. For problems containing no cycles, SAB is time and space linear. MBD1 (similar to GDE [9]) can find all minimal diagnoses and is not restricted by the structure of the problem. MBD2 is an incremental version of MBD1 that focuses on minimal cardinality diagnoses. Since SAB's complexity is time and space linear while MBD1's (MBD2's) is exponential, we expected a significant performance gain by SAB over MBD1 (MBD2). Two sets of experiments are conducted. The first set compares the diagnostic performance of SAB and MBD1 on a class of randomly generated acyclic circuits. The second set compares SAB and MBD2 for diagnosing circuits containing cycles (N-bit adder) with randomly generated inputs and multiple faults. The first experiments show that algorithm SAB is superior to MBD1 for most problem instances. Depending on the circuit type and the input probabilities, there exists a critical circuit size

(in the order of 100 components) beyond which MBD1 becomes inefficient in space and time and thus diagnoses cannot be computed (for at least one generated problem). Unlike MBD1, SAB was able to compute the diagnoses for all circuit sizes studied (3 to 511 components) and all problem instances. In comparison to SAB, MBD1 was markedly sensitive to variations in the probabilities of the inputs and the types of circuits studied. In general, MBD1 seems to be at a disadvantage with respect to SAB when the number of conflict sets is large and the number of minimal-cardinality covers is small. On the other hand, MBD1 is at some advantage when the number of conflicts is small while the number of diagnoses is relatively large. The second set of experiments show that even when relatively expensive, it is worthwhile doing tree clustering once so that all future diagnosis tasks are performed efficiently. The main conclusion is that for nearly acyclic circuits, such as the N-bit adder, the performance of SAB, being linear, provides a definite advantage as the size of a circuit increases and the number of minimal conflicts or diagnoses becomes too large.

References

[1] R. R. Bakker, M. Bourseau, and M. K. de Weger. Pragmatic reasoning in model-based diagnosis. In *Working Notes of the Third International Workshop on Principles of Diagnosis*, pages 107–116, Rosario, WA, USA, 1992.

[2] E. Charniak and S.E. Shimony. Probabilistic semantics for cost-based abduction. In *Proceedings, AAAI-90*, pages 106–111, Menlo Park, CA, 1990. AAAI Press/ The MIT Press.

[3] R. L. Childress and M. Valtorta. Polynomial-time model-based diagnosis with the critical set algorithm. In *Working Notes of the Fourth International Workshop on Principles of Diagnosis*, pages 166 – 177, University of Wales, Aberystwith, 1993.

[4] A. Darwiche. Conditional independence in ATMSs: Independence-based algorithms for computing labels and diagnoses. In *Working Notes of the Fifth International Workshop on Principles of Diagnosis*, pages 70–77, New Paltz, NY, USA, 1994.

[5] J. de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[6] J. de Kleer. Focusing on probable diagnoses. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 842–848, Menlo Park, CA, 1991. AAAI Press/ The MIT Press.

[7] J. de Kleer, A.K. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56:197–222, 1992.

[8] J. de Kleer and O. Raiman. How to diagnose well with very little information. In *Working Notes of the Fourth International Workshop on Principles of Diagnosis*, pages 160–165, University of Wales, Aberystwith, 1993.

[9] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[10] A. Dechter and R. Dechter. Structure-driven algorithms for truth maintenance. *Artificial Intelligence*, 1985.

[11] R. Dechter. Constraint networks. In *Encyclopedia of Artificial Intelligence*, pages 276–285. J. Wiley, New York, second edition, 1992.

[12] R. Dechter and A. Dechter. Belief maintenance in dynamic constraint networks. In *Proceedings, AAAI-88*, pages 37–42, Menlo Park, CA, 1988. AAAI Press/ The MIT Press.

[13] R. Dechter, A. Dechter, and J. Pearl. Optimization in constraint networks. In R.M. Olivier and J.Q. Smith, editors, *Influence Diagrams, Belief Nets and Decision Analysis*, pages 411–425. J. Wiley, New York, 1990.

[14] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.

[15] Y. El Fattah and R. Dechter. Empirical evaluation of diagnosis as optimization in constraint networks. In *Working Notes of the Third International Workshop on Principles of Diagnosis*, pages 149–158, Rosario, WA, USA, 1992.

[16] H. Freitag and G. Friedrich. Focusing on independent diagnosis problems. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 521–531, 1992.

[17] G. Friedrich, G. Gottlob, and W. Nejdl. Physical impossibility instead of fault models. In *Proceedings, AAAI-90*, pages 331–336, Menlo Park, CA, 1990. AAAI Press/ The MIT Press.

[18] H. Geffner and J. Pearl. An improved constraint propagation algorithm for diagnosis. In *Proceedings, IJCAI-87*, pages 1105–1111, 1987.

[19] M. R. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24:411–436, 1984.

[20] D. Maier. *The Theory of Relational Databases.* Computer Science Press, Rockville, MD, 1983.

[21] I. Mozetič. A polynomial-time algorithm for model-based diagnosis. In B. Neumann, editor, *Proceeding of the 10th European Conference on Artificial Intelligence (ECAI 92)*, pages 729–733, New York, 1992. John Wiley & Sons.

[22] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufman, San Mateo, CA, 1988.

[23] Y. Peng and J.A. Reggia. *Abductive Inference Models for Diagnostic Problem Solving.* Springer-Verlag, Berlin, 1990.

[24] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[25] P. Resnick. Generalizing on multiple grounds: Performance learning in model-based troubleshooting. Technical Report AI-TR 1052, MIT Artificial Intelligence Laboratory, February 1989.

[26] E. Santos, Jr. On the generation of alternative explanations with implications for belief revision. In *Proceedings, Uncertainty in Artificial Intelligence*, pages 339–347, 1991.