Journal of Artificial Intelligence Research 51 (2014) 377-411 Submitted 7/14; published 10/14

A Novel SAT-Based Approach to Model Based Diagnosis

Amit Metodi AMITMET@CS.BGU.AC.IL
Department of Computer Science,
Ben Gurion University of the Negev, Beer-Sheva, Israel

Roni Stern RONI.STERN@GMAIL.COM
Meir Kalech KALECH@BGU.AC.IL
Department of Information Systems Engineering,
Ben Gurion University of the Negev, Beer-Sheva, Israel

Michael Codish MCODISH@CS.BGU.AC.IL
Department of Computer Science,
Ben Gurion University of the Negev, Beer-Sheva, Israel

Abstract

This paper introduces a novel encoding of Model Based Diagnosis (MBD) to Boolean Satisfaction (SAT) focusing on minimal cardinality diagnosis. The encoding is based on a combination of sophisticated MBD preprocessing algorithms and the application of a SAT compiler which optimizes the encoding to provide more succinct CNF representations than obtained with previous works. Experimental evidence indicates that our approach is superior to all published algorithms for minimal cardinality MBD. In particular, we can determine, for the first time, minimal cardinality diagnoses for the entire standard ISCAS-85 and 74XXX benchmarks. Our results open the way to improve the state-of-the-art on a range of similar MBD problems.

1. Introduction

Automated diagnosis is concerned with reasoning about the health of systems, including the identification of abnormal behavior, isolation of faulty components and prediction of system behavior under normal and abnormal conditions. As systems become large-scale and more complex, their automated diagnosis becomes more challenging. Model Based Diagnosis (MBD) is an artificial intelligence based approach that aims to cope with the diagnosis problem (Reiter, 1987; de Kleer & Williams, 1987). In MBD, a model of the system is first built. A diagnoser then observes the system to predict its behavior by the model. Discrepancies between the observation and the prediction are used as the input for a diagnosis algorithm which produces a set of possible faults that can explain the observation. MBD has been deployed in several real-world applications, including spacecrafts (Williams & Nayak, 1996), satellite decision support systems (Feldman, de Castro, van Gemund, & Provan, 2013), the automotive industry (Struss & Price, 2003) and spreadsheets (Jannach & Schmitz, 2014). Also, there exist several commercial MBD tools (Feldman, 2012; Dressler & Struss, 1995).

MBD is known to be a hard problem where algorithms have exponential runtime (exponential in the number of components in the diagnosed system). Moreover, the number of potential diagnoses for a given observation can be huge. Therefore, MBD algorithms typically focus on minimal diagnoses: minimal subset – that do not contain other diagnoses, and minimal cardinality – that are smallest in size. Computing the first minimal diagnosis is in P, but computing the next one is NP-hard (Bylander, Allemang, Tanner, & Josephson, 1991). Computing the minimal cardinality is NP-hard, even for the first diagnosis (Selman & Levesque, 1990). In this work we focus on this hard task of finding minimal cardinality diagnoses.

The study of Model-Based Diagnosis has resulted in a variety of computational and modeling challenges. In this paper we focus on one such challenge which has received much attention over the years. Originally defined by Reiter (1987) and by de Kleer and Williams (1987), this problem aims to diagnose multiple faulty components in the so-called *weak fault model*, which ignores the mode of abnormal behavior of components. This problem has been extensively researched for more than 25 years and a wide range of papers propose different algorithms to solve it, including a range of papers from recent years (Feldman & van Gemund, 2006; Williams & Ragno, 2007; Feldman, Provan, & van Gemund, 2010a; Siddiqi & Huang, 2007, 2011). When addressing this challenge, it is common practice to focus on the diagnosis of Combinational Logic Circuits. Namely, Boolean circuits where the single output of each component is determined only by the logical function of its current input state (independent of time and with no feedback).

In the basic setting a diagnosis considers a single observation on the inputs and outputs of the system. Variations consider additional information such as probabilities on component failure, multiple observations on the inputs and outputs of the system, and observations on internal positions of the system ("probes"). In this paper we focus on the basic setting. Extensions and variations are discussed in Section 8.

Even in the basic setting, solving an MBD problem is often impractical, especially for high-cardinality faults. For instance, in a system of 1000 components, to find a minimal cardinality diagnosis of size 5, a diagnosis engine must verify the absence of a diagnosis consisting of 4 components (there are more than 10^{10} such combinations). To overcome the complexity of the problem we consider a novel encoding to SAT.

In recent years, Boolean SAT solving techniques have improved dramatically. Today's SAT solvers are considerably faster and able to manage larger instances than yesterday's. Moreover, encoding and modeling techniques are better understood and increasingly innovative. SAT is currently applied to solve a wide variety of hard and practical combinatorial problems, often outperforming dedicated algorithms. For a survey on the state-of-the-art in SAT solving see the work by Biere, Heule, van Maaren, and Walsh (2009) or the draft of the forthcoming volume of "The Art of Computer Programming" (Knuth, 2014).

The general idea is to encode a (typically, NP) hard problem instance, μ, to a Boolean formula, φ_μ, such that the solutions of μ correspond to the satisfying assignments of φ_μ. Given the encoding, a SAT solver is then applied to solve μ.

SAT-based solutions for MBD have already been proposed. Smith et al. (2005) encode a circuit, representing each component through its clauses and add constraints for cardinality. This is the basis for all the other SAT-based encodings, including the one we contribute in this paper. Bauer (2005) introduces a tailored SAT solver specifically designed to return many diagnoses. Stein et al. (2006) address diagnosis of qualitative models of physical systems with multiple fault modes. More recently, Feldman et al. (2010) propose an encoding to MAX-SAT and demonstrate that off-the-shelf solvers require more calls to a SAT solver than the stochastic diagnosis algorithm SAFARI (Feldman et al., 2010a).

These previous applications of SAT for MBD appear to indicate that SAT and MAX-SAT solvers are doomed to perform poorly on the standard benchmarks (Feldman et al., 2010). This paper proves the contrary. Our SAT-based approach differs from previous SAT encodings in several key aspects. First, sophisticated MBD preprocessing techniques are applied to facilitate the construction of a carefully designed constraint model, which includes constraints that exploit unique substructures in the diagnosed system. Second, this constraint model is compiled to a corresponding CNF using a constraint compiler called BEE (Metodi & Codish, 2012), that simplifies constraints and generates an encoding to CNF which significantly improves the subsequent runtime of the underlying SAT solver. Lastly, a structural abstraction inspired by Siddiqi and Huang (2007) is used to decompose the diagnosis problem, such that the SAT solver is only used to find "top-level" diagnoses, and we show a simple poly-time algorithm to expand these "top-level" diagnoses to find all minimal cardinality diagnoses. Our approach requires some preprocessing of the diagnosed system, but the complexity of that preprocessing is a low-order polynomial, and negligible (both theoretically and empirically) compared to the cost of the actual SAT solving.

We evaluated our SAT-based approach using two standard benchmarks: ISCAS-85 (Brglez, Bryan, & Kozminski, 1989) and 74XXX. These are the standard benchmarks in the MBD literature, used extensively from the time they were made available until today (Feldman & van Gemund, 2006; Feldman et al., 2010a; Siddiqi & Huang, 2007, 2011; Stern, Kalech, Feldman, & Provan, 2012; Nica, Pill, Quaritsch, & Wotawa, 2013). Finding minimal cardinality diagnoses for hard sets of observations in the ISCAS-85 has been a long standing challenge in the MBD community and used in diagnosis competitions (DXC, 2009). The ISCAS-85 systems were also used as a standard for automatic benchmark generation (Wang & Provan, 2010).

We consider three known sets of observations with minimal cardinalities between 1–31, and for the first time succeed to compute a minimal cardinality diagnosis for all observations in the benchmark. We compare our approach to a wide collection of state-of-the-art algorithms for MBD, including: HA* (Feldman & van Gemund, 2006), CDA* (Williams & Ragno, 2007), SAFARI (Feldman et al., 2010a), HDIAG (Siddiqi & Huang, 2007) and DCAS (Siddiqi & Huang, 2011). Results are unequivocal. Our approach outperforms the others, often by orders of magnitude, in terms of runtime. This result is even more significant, as SAFARI is a stochastic algorithm, known as fast, which does not even aim to guarantee minimal cardinality. Our approach, on the other hand, guarantees a minimal cardinality diagnosis and runs faster than SAFARI.

This paper goes beyond our preliminary version of this work (Metodi, Stern, Kalech, & Codish, 2012a). We provide a detailed description of each of the components of our approach, we present detailed algorithms, prove correctness, provide additional examples and present a more elaborate experimental evaluation. In the next section we discuss additional related work. Section 3 presents the required background on MBD. In Section 4 we present the standard approach to model MBD with SAT. Section 5 is the main part of this paper in which we describe the building blocks of our tool to find minimal cardinality diagnosis. Section 6 describes how these building blocks are combined into a diagnosis algorithm. Comprehensive evaluation of our approach is given in Section 7. Section 8 discusses the applicability of our approach in a more general setting and Section 9 concludes.

2. Related Work

Since the late 80's, the Model Based Diagnosis problem with weak fault model has been widely researched and a wide range of papers propose different algorithms to solve it (Reiter, 1987; de Kleer

& Williams, 1987; Feldman & van Gemund, 2006; Williams & Ragno, 2007; Feldman et al., 2010a; Siddiqi & Huang, 2007, 2011). Till today it is considered a challenge as reflected by the "synthetic track" in the annual DXC diagnosis competition (DXC, 2009).

Many of the existing diagnosis techniques propose to apply a combination of deterministic reasoning and search algorithms. One classic approach involves a two stage process. First, it identifies conflict sets, each of which includes at least one fault. Then, it applies a hitting set algorithm to compute sets of multiple faults that explain the observation (de Kleer & Williams, 1987; Williams & Ragno, 2007). These methods guarantee sound diagnoses, and some of them are even complete. However, they tend to fail for large systems due to infeasible runtime or space requirements.

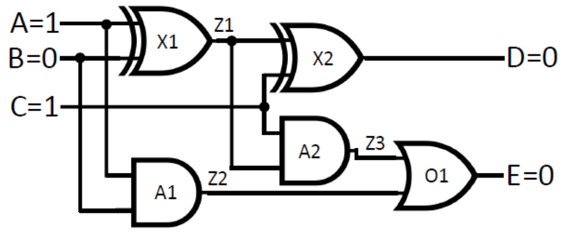
An alternative method is to directly search for diagnoses by trying different assumptions on which components are faulty. For example, the DRUM-II diagnosis engine finds a minimal diagnosis by performing an iterative deepening search, limiting in every iteration, the number of components that are assumed to be faulty (Fröhlich & Nejdl, 1997). DRUM-II also analyzes the dependencies between components to prune irrelevant diagnoses. Recent work presents empirical evidence suggesting that "direct" search for diagnoses is often better than conflict-directed diagnosis algorithms (Nica et al., 2013). Nica et al. did not compare against our SAT-based approach, as it uses pre-processing. In this work we show that the proposed pre-processing is very efficient computationally and results in huge speedups during the search for a diagnosis. This form of pre-processing is a key ingredient which enables us to find very large minimal cardinality diagnoses, even with sizes up to 31.

Another approach considers the diagnosis problem in terms of inductive learning. Here, one tries to learn relations between the symptoms and the faults (Murray, Hughes, & Kreutz-Delgado, 2006). One disadvantage of most works in this approach is that they learn only a single fault rather than multiple faults (Balakrishnan & Honavar, 1998). In addition, inductive learning methods do not guarantee sound diagnoses nor completeness. We, on the other hand, propose a method which addresses multiple faults and guarantees sound and complete minimal cardinality diagnoses.

Feldman et al. (2010a) propose a stochastic diagnosis algorithm, called SAFARI. Although this method is not guaranteed to return diagnoses of minimal cardinality, it presents solutions which are close to minimal cardinality in very low runtime. In Section 7, we demonstrate that our approach outperforms SAFARI in terms of runtime, and also guarantees that minimal cardinality diagnoses are returned.

Compilation-based methods have also been proposed in the MBD context. Torasso and Torta (2006) proposed to compile the system description to a Binary Decision Diagrams (BDDs). Darwiche (2001) proposed to compile the system description into Decomposable Negation Normal Form (DNNF). In both cases, the compiled model allowed finding minimal cardinality diagnosis in time that is polynomial in the size of the compiled model. However, in these works, the size of the compiled model (BDD or DNNF) may grow exponentially and is shown to become a bottleneck (Siddiqi & Huang, 2007).

Siddiqi and Huang (2007) suggest to optimize MBD by identifying components that *dominate* others. We adopt this idea and apply it in our SAT-based approach. Another compilation-based diagnosis algorithm is the HA* algorithm (Feldman & van Gemund, 2006). HA* is designed to exploit a given hierarchy of the diagnosed system. This is done by converting a given system hierarchy to a DNF hierarchy. Each element in this DNF hierarchy is then solved by a simple best-first search using as a heuristic function given a prior probability on the health of the system components. In Section 7, we demonstrate that our approach substantially outperforms HA*.



$COMPS = \{\mathtt{X}_1, \mathtt{X}_2, \mathtt{A}_1, \mathtt{A}_2, \mathtt{O}_1\}$
$OBS = \{\mathtt{A}, \neg\mathtt{B}, \mathtt{C}, \neg\mathtt{D}, \neg\mathtt{E}\}$

Figure 1: An MBD problem: A (faulty) full adder.

Another previously proposed approach imposes a tree structure over a given system description. A system in a tree structure can be diagnosed by joining the diagnoses of its constituent subsystems. El Fattah and Dechter (1995) obtained a tree structure by converting the diagnosed system into a chordal graph and then decomposed it to a tree of maximal cliques. The TREE* algorithm is another such tree-decomposition algorithm, initially proposed only for tree-structured systems (Stumptner & Wotawa, 2001). TREE* was later generalized to perform on any system by embedding a hyper tree over a specific representation of the diagnosed system (Stumptner & Wotawa, 2003). Follow up work further generalized TREE* to support various forms of diagnosis optimization tasks, such as finding minimal cardinality diagnoses, or finding subset minimal diagnoses, or finding most probable diagnoses (Sachenbacher & Williams, 2004). Note that the complexity of TREE* is exponential in the width of the hyper-tree embedded in the system description as defined in these works.

3. Model-Based Diagnosis: Preliminaries

This section introduces the background on Model Based Diagnosis. In addition to the basic definitions, we review several other concepts from the literature that we build on in this paper.

Model Based Diagnosis problems arise when the normal behavior of a system is violated due to faulty components as indicated by certain observations. We focus on *weak fault models*, which ignore the mode of abnormal behavior of components. An MBD problem is specified as a triplet $\langle SD, COMPS, OBS\rangle$ where: SD is a system description, $COMPS$ is a set of components, and OBS is an observation. The system description takes into account that some components might be abnormal (faulty). This is specified by an unary predicate $h(\cdot)$ on components such that $h(c)$ is $true$ when component c is healthy and *false* when c is faulty. Denoting the correct behavior of c as a propositional formula, φ_c, SD is given formally as

$$SD = \bigwedge_{c \in COMPS} h(c) \Rightarrow \varphi_c$$

Namely, each component which is not faulty follows its correct behavior. A diagnosis problem arises when, under the assumption that none of the components are faulty, there is an inconsistency between the system description and the observations (de Kleer & Williams, 1987; Reiter, 1987).

Definition 1 [Diagnosis Problem]. *Given an MBD problem, $\langle SD, COMPS, OBS\rangle$, a diagnosis problem arises when*

$$SD \wedge \bigwedge_{c \in COMPS} h(c) \;\wedge OBS \vdash \bot$$

For example, a diagnosis problem arises for the MBD of Figure 1 as normal behavior would give output $\texttt{E} = \texttt{1}$. Once there is an inconsistency, a diagnosis algorithm tries to find a subset $\Delta \subseteq$ *COMPS* which, if assumed faulty, explains the observation.

Definition 2 [Diagnosis] *Given an MBD problem, $\langle$SD, COMPS, OBS$\rangle$, the set of components $\Delta \subseteq$ COMPS is a* diagnosis *if*

$$\textit{SD} \wedge \bigwedge_{c\in\Delta} \neg h(c) \;\; \wedge \bigwedge_{c\notin\Delta} h(c) \;\wedge\; \textit{OBS} \;\;\nvdash \bot$$

We say that Δ is a minimal diagnosis *if no proper subset $\Delta' \subset \Delta$ is a diagnosis, and that Δ is a* minimal cardinality diagnosis *if no other diagnosis $\Delta' \subseteq$ COMPS exists such that $|\Delta'| < |\Delta|$.*

For the MBD of Figure 1, $\Delta_1=\{\texttt{X}_1, \texttt{X}_2\}$, $\Delta_2=\{\texttt{O}_1\}$, $\Delta_3=\{\texttt{A}_2\}$ are minimal diagnoses, and Δ_2, Δ_3 are minimal cardinality diagnoses, as there is no smaller diagnosis.

An important concept that we make use of in this paper is that of “gate domination”, used for automatic test pattern generation (ATPG) (Kirkland & Mercer, 1987; Fujiwara, Member, Shimono, & Member, 1983) and in some modern SAT solvers (Marques-Silva, Lynce, & Malik, 2009), sometimes under the name “unique sensitization”. Siddiqi and Huang (2007) applied “gate domination” in model-based diagnosis, introducing the notion of a “cone”. The following wording is taken from Siddiqi and Huang’s paper in a setting where the system is a Boolean circuit and the components are its gates.

Definition 3 (Dominator and Cone) *A gate X in the fan-in region of gate G is dominated by G and conversely G is a dominator of X if any path from X to an output of the circuit contains G. The cone corresponding to a gate G is the set of gates dominated by G. A maximal cone is one that is either contained in no other cone or contained in exactly one other cone which is the entire circuit.*

For example, in the circuit depicted as Figure 1, the components $\{\texttt{A}_1, \texttt{A}_2, \texttt{O}_1\}$ form a cone, since any path from $\texttt{A}_1$ or from $\texttt{A}_2$ to a system output contains $\texttt{O}_1$. Here $\texttt{O}_1$ is the dominator and $\texttt{A}_1$ and $\texttt{A}_2$ are the dominated gates.

Although Definition 3 is stated in terms of Boolean circuits and logical gates, the notions of dominators and cones can be generalized for many systems, where components correspond to gates, and a component C_1 dominates a component C_2 if all of the paths passing through C_2 also pass through C_1. For example, in the system illustrated in Figure 3 components C_1 and C_2 form a cone, where C_2 dominates C_1.

The importance of cones to MBD algorithms is rooted in two observations presented by Siddiqi and Huang. Firstly, cones are single-output sub-systems and as such, a minimal cardinality diagnosis will always, independent of the observation, indicate at most one unhealthy component per cone. Secondly, if *C* is a cone in *SD*, then without loss of generality, we may assume that all dominated components in *C* are healthy. This is correct because if *X* is unhealthy in some minimal cardinality diagnosis and dominated by *G*, then *G* must be healthy. So, there exists another minimal cardinality diagnosis where *X* is healthy and *G* is not. For example, in the circuit depicted in Figure 1, Δ_3 is a minimal cardinality diagnosis that signifies dominated $\texttt{A}_2$ as unhealthy, and there exists another minimal cardinality diagnosis, Δ_2, in which $\texttt{A}_2$ is healthy but $\texttt{O}_1$, which dominates $\texttt{A}_2$, is unhealthy.

Based on these observations we can restrict the search for minimal cardinality diagnoses, to so-called “top-level” minimal cardinality diagnoses. The notion of “top-level” diagnoses was introduced by Siddiqi and Huang.

Definition 4 (top-level diagnosis (TLD)) *We say that a minimal cardinality diagnosis is top-level if it does not contain any dominated components.*

To formally justify the focus on top-level diagnoses we make explicit the following Propositions 1 and 2, which are left implicit in previous work.

Proposition 1 *Let Δ' be a minimal cardinality diagnosis for a given MBD problem. Then there is a top-level diagnosis Δ, of the same cardinality.*

Proof: Straightforward. To obtain Δ, replace each dominated component from Δ' by its corresponding dominator. □

We further note that the set of all minimal cardinality diagnoses can be obtained by "expanding" the set of all top-level minimal cardinality diagnoses in the following sense: Given a minimal cardinality top-level diagnosis, $\Delta = \{c_1, \dots, c_\ell\}$ consisting of ℓ dominators from corresponding cones $\{C_1 \dots, C_\ell\}$, denote

$$\chi_i = \left\{ c'_i \in C_i \;\middle|\; \Delta \setminus \{c_i\} \cup \{c'_i\} \text{ is a diagnosis } \right\} \tag{1}$$

We say that Δ expands to the set of minimal cardinality diagnoses defined in terms of a cross-product by: $\gamma(\Delta) = \chi_1 \times \dots \times \chi_\ell$. For example, consider the system from Figure 1 with the observation $\mathit{OBS} = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \neg\mathtt{D}, \neg\mathtt{E}\}$. The cones in the system are $\mathtt{C}_1 = \{\mathtt{X}_1\}$, $\mathtt{C}_2 = \{\mathtt{X}_2\}$, and $\mathtt{C}_3 = \{\mathtt{A}_1, \mathtt{A}_2, \mathtt{O}_1\}$. The corresponding MBD problem has two top-level minimal cardinality diagnoses, $\Delta_1 = \{\mathtt{X}_1, \mathtt{O}_1\}$ and $\Delta_2 = \{\mathtt{X}_2, \mathtt{O}_1\}$ and we have $\gamma(\Delta_1) = \{\mathtt{X}_1\} \times \{\mathtt{O}_1\} = \{\{\mathtt{X}_1, \mathtt{O}_1\}\}$ and $\gamma(\Delta_2) = \{\mathtt{X}_2\} \times \{\mathtt{A}_1, \mathtt{O}_1\} = \{\{\mathtt{X}_2, \mathtt{A}_1\}, \{\mathtt{X}_2, \mathtt{O}_1\}\}$.

Proposition 2 *Δ' is a minimal cardinality diagnosis if and only if there is a top-level minimal cardinality diagnosis Δ that expands to include Δ'.*

Proof: The proof is straightforward from the construction. □

Finally, we comment that the sets χ_i which specify the expansion of a top-level diagnosis Δ in Equation (1) are easy to compute: for each component $c'_i \in C_i$ checking if $\Delta \setminus \{c_i\} \cup \{c'_i\}$ is a diagnosis means propagating the observed inputs through the system, flipping the outputs when propagating through a component in $\Delta \setminus \{c_i\} \cup \{c'_i\}$ and checking if there is no conflict to the observed outputs. This observation is not explicit in previous work and it is essential to justify the focus on top-level diagnoses. Proposition 2 is important as it applies for any diagnosis algorithm. As such, diagnosis algorithms in general can focus, and be compared on finding TLDs, instead of finding all minimal cardinality diagnoses.

4. The Standard Approach to SAT-Based MBD

The standard encoding of an MBD problem $\langle \mathit{SD}, \mathit{COMPS}, \mathit{OBS} \rangle$ to Boolean Satisfiability (as introduced in Smith et al., 2005) associates each component $c \in \mathit{COMPS}$ with a propositional formula, φ_c, denoting its correct behavior, and with a Boolean variable, H_c, signifying if c is "healthy". Viewing the observation as a propositional statement, an encoding is obtained by specifying

$$\varphi = \mathit{OBS} \wedge \bigwedge_{c \in \mathit{COMPS}} H_c \Rightarrow \varphi_c \tag{2}$$

In a satisfying assignment for φ, the health variables assigned the value *false* determine a (not necessarily minimal) diagnosis Δ.

For example, consider the MBD problem of Figure 1, and let $\texttt{comp}(\texttt{A},\texttt{B},\texttt{C})$ with $\texttt{comp} \in \{\texttt{and},\texttt{or},\texttt{xor}\}$ denote the propositional formula describing the behavior of the component which is an "and", "or" or "xor" gate with inputs A, B and output C. So, Equation (2) takes the form:

$$\varphi = \left(\begin{array}{lcll} \texttt{A} \wedge \neg\texttt{B} \wedge \texttt{C} \wedge \neg\texttt{D} \wedge \neg\texttt{E} & \wedge & \texttt{H}_{\texttt{X}_1} \Rightarrow \texttt{xor}(\texttt{A},\texttt{B},\texttt{Z}_1) & \wedge \\ \texttt{H}_{\texttt{A}_1} \Rightarrow \texttt{and}(\texttt{A},\texttt{B},\texttt{Z}_2) & \wedge & \texttt{H}_{\texttt{X}_2} \Rightarrow \texttt{xor}(\texttt{Z}_1,\texttt{C},\texttt{D}) & \wedge \\ \texttt{H}_{\texttt{A}_2} \Rightarrow \texttt{and}(\texttt{Z}_1,\texttt{C},\texttt{Z}_3) & \wedge & \texttt{H}_{\texttt{O}_1} \Rightarrow \texttt{or}(\texttt{Z}_2,\texttt{Z}_3,\texttt{E}) & \end{array} \right) \tag{3}$$

This formula is satisfied by the assignment of variables $\{\texttt{A},\texttt{C},\texttt{H}_{\texttt{A}_1},\texttt{H}_{\texttt{A}_2},\texttt{H}_{\texttt{O}_1}\}$ to *true* and of variables $\{\texttt{B},\texttt{D},\texttt{E},\texttt{Z}_1,\texttt{Z}_2,\texttt{Z}_3,\texttt{H}_{\texttt{X}_1},\texttt{H}_{\texttt{X}_2}\}$ to *false*. This assignment indicates that $\Delta = \{\texttt{X}_1,\texttt{X}_2\}$ is a diagnosis.

To obtain a minimal cardinality diagnosis we seek a satisfying assignment with a minimal number of health variables taking value *false*. For example the assignment of variables $\{\texttt{A},\texttt{C},\texttt{Z}_1,\texttt{Z}_3,\texttt{H}_{\texttt{X}_1},\texttt{H}_{\texttt{X}_2},\texttt{H}_{\texttt{A}_1},\texttt{H}_{\texttt{A}_2}\}$ to *true* and of variables $\{\texttt{B},\texttt{D},\texttt{E},\texttt{Z}_2,\texttt{H}_{\texttt{O}_1}\}$ to *false* which also satisfies Equation (3) and indicates only one faulty component. This can be achieved using a MAX-SAT solver (Feldman et al., 2010), or using a SAT solver as done in the implementation underlying this paper, where a cardinality constraint (encoded to CNF) is introduced to constrain the number of faulty components as detailed in the next section.

No matter how the cardinality constraint is encoded to CNF, for a setting with $|\mathit{COMPS}| = n$ and a constant k, The formula

$$\varphi_k = \varphi \wedge \texttt{sum\_leq}(\left\{ \neg\texttt{H}_\texttt{c} \;\middle|\; \texttt{c} \in \mathit{COMPS} \right\}, \texttt{k}) \tag{4}$$

is satisfied only if at most k of the n health variables take the value *false*. More specifically, we seek a minimal value of k such that (the CNF corresponding to) φ_k is satisfiable. This involves iterating over calls to the SAT solver with formulae φ_k for decreasing values of k until φ_k is satisfiable but φ_{k-1} is not. This approach takes advantage of the fact that SAT solvers are typically incremental: adding clauses to a satisfiable instance allows to solve again while maintaining all of the derived information about the search space from the previous call.

5. Our Approach to SAT-Based MBD

Our approach to encoding an MBD problem $\langle \mathit{SD}, \mathit{COMPS}, \mathit{OBS} \rangle$ to SAT proceeds as follows: First, we adopt a finite domain constraint based representation to express the basic model. Second, we analyze the structure and substructures of the *SD* to introduce additional (redundant) constraints that will later boost the search for a minimal cardinality analysis. Third, we introduce constraints to model the given observation *OBS* with an additional constraint that imposes a bound on the cardinality of the diagnosis (the number of unhealthy components). This additional constraint reduces the subsequent number of iterations in search of the minimal cardinality diagnosis. Each such iteration involves a call to the underlying SAT solver and hence has worst-time exponential complexity. So, reducing this number is important. Given all of these constraints, we apply a finite domain constraint compiler (Metodi & Codish, 2012; Metodi, Codish, & Stuckey, 2013) to simplify and encode them to a corresponding CNF. Finally we apply a SAT solver to seek a suitable satisfying assignment and solve the problem. In the rest of this section we describe these phases in more detail. An

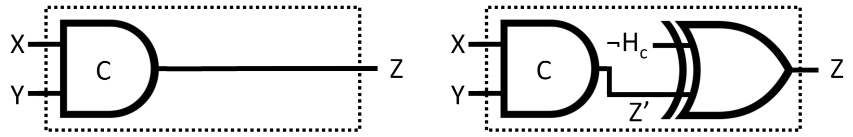


Figure 2: Modeling component c by composition with `xor`

experimental evaluation illustrating the impact of the various constraints in the model is presented in Section 7.

5.1 The Basic Model for MBD

We build on the standard approach, as in Equation (2). However, we observe that for model based diagnosis in the weak fault model with a single observation and when searching for a minimal diagnosis, the behavior of a faulty component can be assumed to produce an output opposite to that of its normal behavior. This is because any diagnosis that assumes that a component c is faulty but still produces its normal output can be replaced by a smaller diagnosis that does not contain c. Thus, if Δ is a minimal diagnosis (i.e, no subset of Δ is a diagnosis), this means that all the components in Δ are assumed to produce the opposite of their normal output. In this paper we focus on minimal cardinality diagnoses, which are in particular also minimal subset, so we modify Equation (2) as follows, replacing the implication by a bi-implication.

$$\varphi = \mathit{OBS} \ \wedge \bigwedge_{c \in \mathit{COMPS}} H_c \Leftrightarrow \varphi_c \tag{5}$$

We model the behavior ($H_c \Leftrightarrow \varphi_c$) of a possibly faulty component c as if encapsulated together with a `xor` gate as illustrated in Figure 2. Here, the output of the encapsulated component is the `xor` of the usual output of c and its negated health variable $\neg H_c$. One can observe that if H_c is $true$ then this composition is equivalent to the normal behavior of c, otherwise it is equivalent to component c with a negated output.

Our decision to model the relation between a component c and its health variable H_c by introducing an additional `xor` gate (instead of just introducing CNF clauses to directly encode $H_c \Leftrightarrow \varphi_c$) has two motivations: (1) to improve CNF encodings we provide tools to reason about, and simplify system components — so there is an advantage to a uniform representation where all of the logic is expressed in the system model itself; and (2) the underlying SAT solver that we apply, CryptoMiniSat (Soos, 2010), offers direct support for `xor` clauses. Because of (1) our MBD problem is more amenable to simplification, and because of (2) the underlying SAT solver can optimize the search for a satisfying assignment. We comment that it is straightforward to apply our technique with other SAT solvers, which do not support `xor` clauses, by adding their CNF encodings to the model. The finite domain constraint compiler, BEE (Metodi & Codish, 2012), which we apply, is configurable to work with both types of solvers.

As in Equation (3), we write $\mathtt{comp(A,B,C)}$ with $\mathtt{comp} \in \{\mathtt{and}, \mathtt{or}, \mathtt{xor}\}$ to represent a component which is an "$\mathtt{and}$", "$\mathtt{or}$" or "$\mathtt{xor}$" gate with inputs $\mathtt{A,B}$ and output $\mathtt{C}$. We also write $\mathtt{comp_H(A,B,C)}$ to represent the corresponding encapsulated component with a health variable $\mathtt{H}$. So,

$$\mathtt{comp_H(A,B,C)} = \mathtt{comp(A,B,C')} \wedge \mathtt{xor}(\neg\mathtt{H},\mathtt{C'},\mathtt{C})$$

and we view

$$\boxed{\mathtt{comp}_{\mathtt{H}}(\mathtt{A},\mathtt{B},\mathtt{C})} \qquad \text{(Constraints 1)}$$

as a constraint on the Boolean variables $\mathtt{A},\mathtt{B},\mathtt{C}$ and $\mathtt{H}$. Given this notation, the system depicted as Figure 1 is modeled by the following constraints:

$$\begin{array}{lllll}\mathtt{xor}_{\mathtt{H}_{\mathtt{X}_1}}(\mathtt{A},\mathtt{B},\mathtt{Z}_1) & \wedge & \mathtt{and}_{\mathtt{H}_{\mathtt{A}_1}}(\mathtt{A},\mathtt{B},\mathtt{Z}_2) & \wedge & \mathtt{xor}_{\mathtt{H}_{\mathtt{X}_2}}(\mathtt{Z}_1,\mathtt{C},\mathtt{D}) \quad \wedge \\ \mathtt{and}_{\mathtt{H}_{\mathtt{A}_2}}(\mathtt{Z}_1,\mathtt{C},\mathtt{Z}_3) & \wedge & \mathtt{or}_{\mathtt{H}_{\mathtt{O}_1}}(\mathtt{Z}_2,\mathtt{Z}_3,\mathtt{E}) & & \end{array}$$

Finally, we add to the constraints representing the system components an additional cardinality constraint:

$$\boxed{\mathtt{sum\_leq}(\left\{ \neg\mathtt{H}_{\mathtt{c}} \;\middle|\; \mathtt{c} \in \mathit{COMPS} \right\}, \mathtt{k})} \qquad \text{(Constraint 2)}$$

to specify for an integer constant k that the number of faulty components must be at most k. For example, for the system depicted as Figure 1 and a constant k, we introduce the constraint $\mathtt{sum\_leq}(\{\neg\mathtt{H}_{\mathtt{X}_1}, \neg\mathtt{H}_{\mathtt{X}_2}, \neg\mathtt{H}_{\mathtt{A}_1}, \neg\mathtt{H}_{\mathtt{A}_2}, \neg\mathtt{H}_{\mathtt{O}_1}\}, \mathtt{k})$. Later we will require to satisfy the constraints of the model and also to minimize the value of k.

To summarize this presentation of the basic model, we show the complete constraint model for the minimal cardinality diagnosis of the MBD problem of Figure 1. For an integer value k, a solution of these constraints is a diagnosis of cardinality $\leq k$:

$$\begin{array}{l}\mathtt{xor}_{\mathtt{H}_{\mathtt{X}_1}}(\mathtt{A},\mathtt{B},\mathtt{Z}_1) \;\wedge\; \mathtt{and}_{\mathtt{H}_{\mathtt{A}_1}}(\mathtt{A},\mathtt{B},\mathtt{Z}_2) \;\wedge\; \mathtt{xor}_{\mathtt{H}_{\mathtt{X}_2}}(\mathtt{Z}_1,\mathtt{C},\mathtt{D}) \;\wedge \\ \mathtt{and}_{\mathtt{H}_{\mathtt{A}_2}}(\mathtt{Z}_1,\mathtt{C},\mathtt{Z}_3) \;\wedge\; \mathtt{or}_{\mathtt{H}_{\mathtt{O}_1}}(\mathtt{Z}_2,\mathtt{Z}_3,\mathtt{E}) \;\wedge \\ \mathtt{sum\_leq}(\{\neg\mathtt{H}_{\mathtt{X}_1}, \neg\mathtt{H}_{\mathtt{X}_2}, \neg\mathtt{H}_{\mathtt{A}_1}, \neg\mathtt{H}_{\mathtt{A}_2}, \neg\mathtt{H}_{\mathtt{O}_1}\}, \mathtt{k}) \wedge \\ \mathtt{A} = \mathtt{1} \;\wedge\; \mathtt{B} = \mathtt{0} \;\wedge\; \mathtt{C} = \mathtt{1} \;\wedge\; \mathtt{D} = \mathtt{0} \;\wedge\; \mathtt{E} = \mathtt{0}\end{array}$$

This type of constraint model can be solved by encoding it to a CNF formula and then applying a SAT solver. By repeatedly seeking a solution for decreasing values of k we can find a minimal cardinality diagnosis. However, we do not apply this basic modeling. Instead we further refine it as described in the rest of this section.

5.2 Encoding Cardinality Constraints

The encoding of cardinality constraints to CNF is the topic of a large body of research papers. Many of these, such as that described by Eén and Sörensson (2006), are based on the use of Batcher's odd-even sorting network (Batcher, 1968). A sorting network is a Boolean circuit with n inputs and n outputs. Given Boolean values on its inputs, the output consists of the same values but sorted: say, zeroes before ones. In our context we apply such a sorting network where the n inputs are the health variables of the n components in the given system, and the n outputs are the sorted values. Now, to encode that at most k health variables take the value *false* we assert that the $(k+1)^{th}$ output of the sorting network is a one. Because its outputs are sorted this implies that the last $n-k$ outputs are also ones thus imposing that "at most" all of the k remaining outputs are zero. Looking backwards through the sorting network this implies that at most k of its inputs take the value *false*.

Sorting networks, like other Boolean circuits are straightforward to encode to a CNF formula. With Batcher's odd-even construction this results in a CNF with $O(n\log^2(n))$ clauses. Further improvements enable an encoding with $O(n\log^2(k))$ clauses to constrain the sum of the n Boolean inputs to be less than k (Asín, Nieuwenhuis, Oliveras, & Rodríguez-Carbonell, 2009, 2011; Codish

& Zazon-Ivry, 2010). In this paper we encode cardinality constraints to CNF using BEE (Metodi & Codish, 2012; Metodi et al., 2013) which takes such an improved approach.

5.3 From Cones to Sections

Reasoning about relations between the components in a system description *SD* enables to infer additional constraints on the number of unhealthy components in certain subsystems of *SD*. These constraints when compiled into the CNF, help boost the search, by the SAT solver, for a minimal cardinality diagnosis. Proposition 2 enables a diagnosis algorithm to focus on top-level diagnoses based on a partitioning of the system into cones: each cone contains at most one unhealthy component, and without loss of generality, it can be assumed to be the dominator of the cone.

To restrict the SAT-based search to top-level minimal cardinality diagnoses we simply add the following constraints where D denotes the set of dominated components.

$$\boxed{\bigwedge_{\mathtt{c}\in\mathtt{D}} \mathtt{H}_{\mathtt{c}}} \qquad \text{(Constraint 3)}$$

Introducing constraints to indicate healthy components reduces the number of (unassigned) health variables and hence boosts the search for minimal cardinality diagnosis. In Section 7 we show that reasoning about cones to restrict the search to top-level diagnoses improves considerably the search for minimal cardinality diagnosis.

Motivated by the utility of partitioning a system into cones, we seek a more general partitioning, which enables to apply similar cardinality constraints to larger subsystems of components. To this end we introduce the notion of a "section". We denote by $sysout(c)$ the set of system outputs which occur at the end of a path from a component c. As an example, in the system depicted in Figure 3 $sysout(\mathtt{C}_1) = \{\mathtt{O}_2, \mathtt{O}_3\}$ and $sysout(\mathtt{C}_5) = \{\mathtt{O}_1, \mathtt{O}_2\}$.

Definition 5 (Section) *Given a system description SD with components COMPS we define a disjoint partitioning COMPS* $= S_1 \cup S_2 \cup \cdots \cup S_n$ *such that for every* $c_1, c_2 \in$ *COMPS,* c_1 *and* c_2 *are in the same section* S_i *if and only if* $sysout(c_1) = sysout(c_2)$.

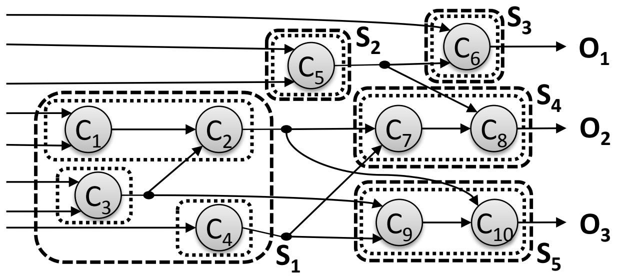


Figure 3: Partitioning a system into cones and sections.

Figure 3 shows a partitioning of a system into maximal cones and sections. The cones are depicted with dotted lines, and the sections with dashed. For example, components $\{\mathtt{C}_1, \mathtt{C}_2\}$ form a cone, and section $\mathtt{S}_1$ consists of three cones. We observe that partitioning a system into sections can be done in polynomial time as demonstrated by Algorithm 1 presented below. Given a partitioning

$\{\mathtt{S_1}, \ldots, \mathtt{S_n}\}$ to sections, we introduce to the constraint model the following constraints which further improve the encoding and hence the subsequent search for minimal cardinality diagnosis. For each section $\mathtt{S_i}$, the constraint

$$\boxed{\mathtt{sum\_leq}(\left\{ \neg\mathtt{H_c} \;\middle|\; \mathtt{c} \in \mathtt{S_i} \right\}, \mathtt{b_i})} \qquad \text{(Constraints 4)}$$

expresses that the sum of the negated health variables in $\mathtt{S_i}$ is bounded from above by a constant $\mathtt{b_i}$ that is the smaller of the following two bounds on the number of unhealthy components in section $\mathtt{S_i}$: (a) The number of outputs from $\mathtt{S_i}$; and (b) The value of $|sysout(\mathtt{c})|$ for some component $\mathtt{c} \in \mathtt{S_i}$. Note that by Definition 5, this value is the same for any $\mathtt{c} \in \mathtt{S_i}$. We justify this statement below in Proposition 3.

To illustrate the utility of sections, consider again the system given as Figure 3 and its partition into 5 sections. Observe that the section labeled $\mathtt{S_1}$ has 3 outputs, but each component $\mathtt{c} \in \mathtt{S_1}$ has only 2 corresponding system outputs ($|sysout(\mathtt{c})| = 2$). So, $\mathtt{b_1} = min\{3, 2\}$ and hence 2 is an upper bound on the number of unhealthy components in $\mathtt{S_1}$. This is also an improvement over the reasoning with cones where the bound on the number of unhealthy components in $\mathtt{S_1}$ is 3 (since there are three cones). Similarly, $\mathtt{b_2} = min\{1, 2\}$, $\mathtt{b_3} = min\{1, 1\}$, $\mathtt{b_4} = min\{1, 1\}$ and $\mathtt{b_5} = min\{1, 1\}$.

Reasoning about constraints on the number of faulty components per section facilitates the MBD encoding in another way. For Constraints 4 we have already encoded the number of faulty components per section. These numbers are partial sums in the context of Constraint 2 which specifies the total number of faulty components in the system and can be reused in the encoding.

The following justifies Constraints 4 .

Proposition 3 *Let $\langle$SD, COMPS, OBS$\rangle$ be an MBD problem, $S \subseteq$ SD be a section, $c \in S$ be a component and Δ be a minimal cardinality diagnosis. Then, both (a) the number of outputs from S, and (b) the value $|sysout(c)|$ are bounds on the number of unhealthy components (from Δ) in S.*

Proof: The statement regarding the number of outputs from S follows directly from an assertion by de Kleer (2008) that the number of outputs from any (sub-) system is a bound on the number of its unhealthy components. So, it remains to prove the statement regarding $|sysout(c)|$.

Assume the premise of the proposition, denote $|\Delta| = k$ and $|S \cap \Delta| = t$ (so $t \leq k$). Assume for contradiction that $t > |sysout(c)|$. We construct a diagnosis Δ' with less than k unhealthy components. First note the obvious: that given Δ we can propagate the observed system inputs to the system outputs where in each step we choose a component with known inputs and produce its normal output if the component is healthy, or its opposite to normal output otherwise. Because Δ is a diagnosis this process will result in no contradictions between propagated outputs and the observed outputs.

Now, take $\Delta' = \Delta \setminus S$. This is not (yet) a diagnosis. With Δ', propagate the observed system inputs in the same way as before with Δ. Now, because Δ' is not a diagnosis there will be some "flipped" system outputs (those which contradict the observed outputs). Each such "flipped" output o must be due to one of the unhealthy components in S that was marked healthy in Δ' and so we have $o \in sysout(c)$. Now consider the component g which outputs o. If $g \in \Delta'$, then remove it; and if $g \notin \Delta'$, then add it. So, now Δ' is a diagnosis and $k' = |\Delta'| \leq k - t + |sysout(c)| < k$. $\square$

Algorithm 1 describes how to partition a system into sections. Denoting the components and outputs of the system as COMPS $= \{c_1, \ldots, c_n\}$ and OUTS $= \{o_1, \ldots, o_m\}$, an $n \times m$ Boolean matrix $\mathbf{b}$ is computed so that $b_{ij} = true$ if $o_j \in sysout(c_i)$ and $false$ otherwise. Figure 4 shows an example of this matrix $\mathbf{b}$ for the system in Figure 3. So, by Definition 5, a pair of components c_i, c_j are in the same section if and only if row i and row j in matrix $\mathbf{b}$ are identical. For instance, section $\mathtt{S}_5$ includes the components $\mathtt{C}_9$ and $\mathtt{C}_{10}$ since their system output $\mathtt{O}_3$ is identical. The computational complexity of this partitioning process is the complexity of running a graph search algorithm for every system output and is in the worst case $O(n^2 \cdot m)$. The algorithm returns a mapping from components to bit vectors which can be seen as section identifiers. So, the algorithm returns a mapping of components to sections.

Algorithm 1 partitioning a system into sections
input: A system (view it as a graph)
output: A partitioning of the system into sections

```
1: Denote: $COMPS$ $=$ $\{c_1, \dots, c_n\}$   the system components
           $OUTS$  $=$ $\{o_1, \dots, o_m\}$   the system outputs
           $\mathbf{b}$ $=$ $(b_{ij})$           an $n \times m$ Boolean matrix
2: for all ($o_j \in OUTS$) do
3:     apply DFS on the reverse edges of the system, with $source = o_j$
4:     for all ($c_i \in COMPS$) do
5:         $b_{ij} = (c_i$ is reachable from $o_j)$
6: return $\{\ c_i \mapsto \langle b_{i1}, \dots, b_{im} \rangle \mid 1 \le i \le n\ \}$
```

Example 1 (partition into sections) *Consider the (abstract) system depicted as Figure 3 where* $COMPS = \{c_1, \dots, c_{10}\}$ *and* $OUTS = \{o_1, \dots, o_3\}$. *The Boolean matrix evaluated by application of Algorithm 1 is illustrated in Figure 4.*

$$\mathbf{b} = \left(\begin{array}{ccc} 0&1&1\\ 0&1&1\\ 0&1&1\\ 0&1&1\\ 1&1&0\\ 1&0&0\\ 0&1&0\\ 0&1&0\\ 0&0&1\\ 0&0&1 \end{array}\right) \begin{array}{l} \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\} \mathtt{S}_1 \\ \left.\begin{array}{l} \\ \end{array}\right\} \mathtt{S}_2 \\ \left.\begin{array}{l} \\ \end{array}\right\} \mathtt{S}_3 \\ \left.\begin{array}{l} \\ \\ \end{array}\right\} \mathtt{S}_4 \\ \left.\begin{array}{l} \\ \\ \end{array}\right\} \mathtt{S}_5 \end{array}$$

Figure 4: Partitioning the system from Figure 3 into sections

There is another benefit of partitioning into sections: the identification of cones may be performed "per section" which is more efficient. This works because, if component X is dominated by component G then $sysout(X) = sysout(G)$ implying that the components of a cone are always in the same section. For example, in Figure 3 component $\mathtt{C}_1$ is dominated by $\mathtt{C}_2$ and $sysout(\mathtt{C}_1) = sysout(\mathtt{C}_2) = \{\mathtt{O}_1, \mathtt{O}_2\}$.

The recursively defined Algorithm 2 shows how to compute cones given a partition into sections. It computes the set of dominators for a component c in a section S of the system. We denote by $succ(c)$ the set of components that c feeds into directly. If $c \in S$ feeds into a component that is not in S then it is only dominated by itself. Otherwise, c is dominated by $c' \in S$ only if c' is dominated by all elements of $succ(c)$. For instance, given section $\mathtt{S}_1$ and component $\mathtt{C}_1$ in Figure 3 $succ(C_1) = \{C_2\}$. In the next recursive call $succ(C_2)$ does not include any component of $\mathtt{S}_1$

(condition in line 2) and thus $\mathtt{C}_2$ is returned (line 5). The union of both calls $(\mathtt{C}_1, \mathtt{C}_2)$ is returned as a cone (line 3).

It is straightforward to implement Algorithm 2 efficiently using a memoization table to avoid recomputing dominators for components already encountered. Since a system is a directed acyclic graph, the recursion in Algorithm 2 will halt when a "leaf" node is reached. Thus the complexity of calculating the dominators of every component c in a section S is $O(|S|^2)$. Given the sets of dominators per component, it is straightforward to specify the set of maximal cones. A component c is a dominator of a maximal cone, if it is only dominated by itself, and the maximal cone corresponding to such a c is the set of components which have c as dominator.

To find all cones in a system, Algorithm 2 is applied once per component per section, and the cost depends on the size of the largest section. In contrast, without the partition into sections, the same algorithm is applied, but considering all of the components in the system instead of all the components in a section. Practice shows that the partition into sections benefits the computation of cones.

Algorithm 2 dominators (component c, section S, system C)

input: component c in section S of system C

output: The set of dominators of c

```
1: Denote: $succ(c) = \left\{ c' \in C \,\middle|\, \text{the output of } c \text{ is an input to } c' \right\}$
2: if $(succ(c) \subseteq S)$ then
3:     return  $\{c\} \cup \bigcap_{c' \in succ(c)}$ dominators$(c', S, C)$
4: else
5:     return $\{c\}$
```

5.4 Modeling the Observation and Further Boosting the Search

Let OBS^+ and OBS^- denote the sets of variables assigned *true* and *false* in OBS, respectively. Then, to model the observation we add the obvious constraints.

$$\boxed{\bigwedge_{\mathtt{x} \in \mathit{OBS}^+} \mathtt{x} \;\wedge\; \bigwedge_{\mathtt{x} \in \mathit{OBS}^-} \neg\mathtt{x}} \qquad \text{(Constraint 5)}$$

To improve the search for a minimal cardinality diagnosis one can introduce an upper bound on the minimal cardinality: the number of outputs in a system is an upper bound on the minimal cardinality (de Kleer, 2008). Such a bound, as well as the section-specific constraint given above (Constraints 4) "ignores" the observed inputs and outputs. Siddiqi and Huang (2011) propose to obtain a tighter upper bound on the minimal cardinality for a given observation by propagating the input values through the system, and taking as an upper bound the number of contradictions between the observed and the propagated outputs. For example, considering the MBD problem from Figure 1, $k = 2$ is an upper bound on the size of a minimal cardinality diagnosis because the system has 2 outputs. Siddiqi and Huang's proposal states that also 1 is an upper bound because when propagating the inputs through the system there is only one contradiction to the observed outputs.

While Siddiqi and Huang's (2011) proposal is intuitively appealing, it is correct only in case that no observed output is also input to another component, and in fact their results are restricted to systems where this is the case. This does not work for the example in Figure 5. Propagating the

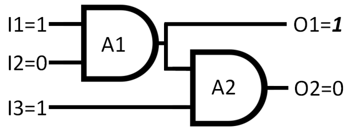


Figure 5: Minimal cardinality diagnosis is of size 2, but propagating observed inputs leads to 1 contradiction to the observed outputs.

observed inputs through the system assigns 0 to both outputs indicating a single contradiction with the observation (on $\mathtt{O_1}$). However, the smallest diagnosis for this example has cardinality 2. This example is not contrived: 83 of the 350 observations for system `74181` of the 74XXX benchmark, exhibit a minimal cardinality diagnosis larger than the (erroneous) bound obtained when counting conflicts between propagated and observed outputs.

Algorithm 3 is about computing an upper bound on the number of faulty components by propagating observed inputs and counting conflicts. The algorithm computes a diagnosis Δ, and $|\Delta|$ is thus an upper bound on the minimal cardinality of a diagnosis. The basic idea is to propagate inputs as long as they do not contradict observed, or other already computed, outputs. The components in the system are processed one at a time. At line 3 we select some component c whose inputs are already determined (initially only the system inputs are determined). For this c we consider its 'already determined' output, o_{obs}, and denote $o_{obs} = \perp$ if its output is not yet determined. We also consider its propagated output o_{prop} which is obtained by propagating the inputs through c assuming that c is healthy. Now there are three cases (lines 6–10): if c has no already determined output then we fix its output to o_{prop} and mark c as healthy. If c has an already determined output and it is consistent with o_{prop} then we also mark c as healthy. Otherwise we mark c as not healthy, but do not propagate its output (which is already determined).

Algorithm 3 Find a diagnosis Δ (and upper bound $|\Delta|$ on min. card.)
input: A system with components, *COMPS*, and observation, *OBS*
output: A diagnosis Δ

```
1: $C \leftarrow \mathit{COMPS}, \ \ \Delta = \emptyset$
2: while $(C \neq \emptyset)$ do
3:     select $c \in C$ such that the inputs of $c$ are determined
4:     $o_{obs} \leftarrow$ the value on the output of $c$ (N/A if it is undefined)
5:     $o_{prop} \leftarrow$ the value if propagating the inputs of $c$ (assume $c$ is healthy)
6:     if $(o_{obs} = \mathrm{N/A}$ then
7:         set output of $c$ to $o_{prop}$ and mark $c$ as healthy
8:     else if $(o_{obs} = o_{prop})$ then
9:         mark $c$ as healthy
10:    else
11:        mark $c$ as faulty and $\Delta = \Delta \cup \{c\}$
12:    $C \leftarrow C \setminus \{c\}$
13: Return $\Delta$
```

When Algorithm 3 terminates we have marked all components as healthy or faulty and have in fact determined a correct diagnosis. As such applying Algorithm 3 provides an upper bound on a minimal cardinality diagnosis – the number of components marked as faulty in the returned diagnosis. Note that Algorithm 3 is correct also when given probes (observed values on the outputs from internal components). Assuming that the components are maintained in a data-structure where components are sorted (topologic) according to their depth, Algorithm 3 is performed as a single linear traversal of this data-structure with complexity $O(|\mathit{COMPS}|)$.

As an example application of Algorithm 3, consider the circuit in Figure 5. Propagating the inputs of gate $\mathtt{A}_1$ gives the output $\mathtt{0}$ in contradiction to the observation on $\mathtt{O}_1$. Hence, we mark $\mathtt{A}_1$ as unhealthy and propagate the observation $\mathtt{O}_1 = \mathtt{1}$ as an input to $\mathtt{A}_2$ together with $\mathtt{I}_3 = \mathtt{1}$. This results in an additional contradiction to the observation $\mathtt{O}_2 = \mathtt{0}$ and so we mark $\mathtt{A}_2$ as unhealthy too, and report $\Delta = \{\mathtt{A}_1, \mathtt{A}_2\}$ hence the value 2 as an upper bound for the minimal cardinality.

Let k_{UB} be the bound found by application of Algorithm 3. We refine Constraint 2 and introduce instead:

$$\boxed{\mathtt{sum\_leq}(\left\{ \neg \mathtt{H_c} \;\middle|\; \mathtt{c} \in \mathit{COMPS} \right\}, \mathtt{k_{UB}})} \qquad \text{(Constraint 2}'\text{)}$$

To appreciate the impact of Algorithm 3 we note that, for the benchmark considered in this paper, Algorithm 3 determines an upper bound equal to the actual minimal cardinality for 81% of the 28,527 observations considered. Of course, even when given a precise upper bound, an MBD algorithm still needs to validate its minimality. In our SAT-based approach this requires one single iteration with the underlying SAT solver. Typically, this is the hardest iteration as it involves a call which is unsatisfiable and is of the largest cardinality for which the instance is unsatisfiable.

5.5 Compiling Constraints to CNF

Metodi and Codish (2012) introduced a compiler called BEE that encodes finite domain constraints to CNF. Besides facilitating the encoding process, this compiler also applies partial evaluation and other optimizations to simplify the constraints before encoding them to CNF. In particular, it applies "equi-propagation" (Metodi, Codish, Lagoon, & Stuckey, 2011) which is the process of identifying equalities between literals (and constants) implied by other such equations and a given constraint. If $\mathtt{X}{=}\mathtt{L}$ is implied by a constraint (where $\mathtt{X}$ is a variable and $\mathtt{L}$ is a literal or a Boolean constant), then all occurrences of $\mathtt{X}$ can be replaced by $\mathtt{L}$, reducing the number of variables in the subsequent CNF encoding. We illustrate constraint simplification for the diagnosis of the circuit in Figure 1. Consider the following constraints (we have omitted some of the constraints as they do not contribute to the example):

$$\begin{array}{ll}
(\mathbf{1}) & \mathtt{xor}_{\mathtt{H_{X_1}}}(\mathtt{A},\mathtt{B},\mathtt{Z_1}) \;\wedge\; \mathtt{and}_{\mathtt{H_{A_1}}}(\mathtt{A},\mathtt{B},\mathtt{Z_2}) \;\wedge\; \mathtt{xor}_{\mathtt{H_{X_2}}}(\mathtt{Z_1},\mathtt{C},\mathtt{D}) \;\wedge \\
(\mathbf{2}) & \mathtt{and}_{\mathtt{H_{A_2}}}(\mathtt{Z_1},\mathtt{C},\mathtt{Z_3}) \;\wedge\; \mathtt{or}_{\mathtt{H_{O_1}}}(\mathtt{Z_2},\mathtt{Z_3},\mathtt{E}) \;\wedge \\
(\mathbf{3}) & \mathtt{sum\_leq}(\{\neg\mathtt{H_{X_1}},\neg\mathtt{H_{X_2}},\neg\mathtt{H_{A_1}},\neg\mathtt{H_{A_2}},\neg\mathtt{H_{O_1}}\},\mathtt{k}) \wedge \\
(\mathbf{4}) & \mathtt{A}=\mathtt{1} \;\wedge\; \mathtt{B}=\mathtt{0} \;\wedge\; \mathtt{C}=\mathtt{1} \;\wedge\; \mathtt{D}=\mathtt{0} \;\wedge\; \mathtt{E}=\mathtt{0} \wedge \\
(\mathbf{5}) & \mathtt{H_{A_1}}=\mathtt{1} \;\wedge\; \mathtt{H_{A_2}}=\mathtt{1}
\end{array}$$

The constraints on lines $(\mathbf{1}) - (\mathbf{3})$ comprise the basic constraint model described in Section 5.1; the constraints on line $(\mathbf{4})$ model the observation; and the constraints in line $(\mathbf{5})$ express that without loss of generality the dominated components $\{\mathtt{A}_1, \mathtt{A}_2\}$ from the cone $\{\mathtt{A}_1, \mathtt{A}_2, \mathtt{O}_1\}$ are healthy. We observe the following equi-propagation steps:

1. $(\mathtt{A}=\mathtt{1}) \wedge (\mathtt{B}=\mathtt{0}) \wedge \mathtt{xor}_{\mathtt{H_{X_1}}}(\mathtt{A},\mathtt{B},\mathtt{Z_1}) \models (\mathtt{Z_1}=\mathtt{H_{X_1}})$

2. $(\mathtt{A}=\mathtt{1}) \wedge (\mathtt{B}=\mathtt{0}) \wedge (\mathtt{H_{A_1}}=\mathtt{1}) \wedge \mathtt{and}_{\mathtt{H_{A_1}}}(\mathtt{A},\mathtt{B},\mathtt{Z_2}) \models (\mathtt{Z_2}=\mathtt{0})$

3. $(\mathtt{C}=1) \wedge (\mathtt{H}_{\mathtt{A}_2}=1) \wedge \mathtt{and}_{\mathtt{H}_{\mathtt{A}_2}}(\mathtt{Z}_1,\mathtt{C},\mathtt{Z}_3) \models (\mathtt{Z}_1=\mathtt{Z}_3)$
4. $(\mathtt{C}=1) \wedge (\mathtt{D}=0) \wedge \mathtt{xor}_{\mathtt{H}_{\mathtt{X}_2}}(\mathtt{Z}_1,\mathtt{C},\mathtt{D}) \models (\mathtt{Z}_1=\mathtt{H}_{\mathtt{X}_2})$
5. $(\mathtt{E}=1) \wedge (\mathtt{Z}_2=0) \wedge \mathtt{or}_{\mathtt{H}_{\mathtt{O}_1}}(\mathtt{Z}_2,\mathtt{Z}_3,\mathtt{E}) \models (\mathtt{Z}_3=\neg\mathtt{H}_{\mathtt{O}_1})$

From these (and the other given) equalities between literals we obtain a substitution:

$$\theta = \left\{ \begin{array}{l} \mathtt{A}\mapsto 1, \mathtt{B}\mapsto 0, \mathtt{C}\mapsto 1, \mathtt{D}\mapsto 0, \mathtt{E}\mapsto 0, \mathtt{Z}_1\mapsto \mathtt{H}_{\mathtt{X}_1}, \mathtt{Z}_2\mapsto 0, \\ \mathtt{Z}_3\mapsto \mathtt{H}_{\mathtt{X}_1}, \mathtt{H}_{\mathtt{X}_2}\mapsto \mathtt{H}_{\mathtt{X}_1}, \mathtt{H}_{\mathtt{A}_1}\mapsto 1, \mathtt{H}_{\mathtt{A}_2}\mapsto 1, \mathtt{H}_{\mathtt{O}_1}\mapsto \neg\mathtt{H}_{\mathtt{X}_1} \end{array} \right\}$$

Applying θ to specialize the constraint system we get:

$$\begin{array}{lllll} (\mathbf{1}) & \mathtt{xor}_{\mathtt{H}_{\mathtt{X}_1}}(1,0,\mathtt{H}_{\mathtt{X}_1}) \wedge & \mathtt{and}_1(1,0,0) & \wedge & \mathtt{xor}_{\mathtt{H}_{\mathtt{X}_1}}(\mathtt{H}_{\mathtt{X}_1},1,0) \wedge \\ (\mathbf{2}) & \mathtt{and}_1(\mathtt{H}_{\mathtt{X}_1},1,\mathtt{H}_{\mathtt{X}_1}) \wedge & \mathtt{or}_{(\neg\mathtt{H}_{\mathtt{X}_1})}(0,\mathtt{H}_{\mathtt{X}_1},0) & \wedge & \\ (\mathbf{3}) & \multicolumn{4}{l}{\mathtt{sum\_leq}(\{\neg\mathtt{H}_{\mathtt{X}_1},\neg\mathtt{H}_{\mathtt{X}_1},0,0,\mathtt{H}_{\mathtt{X}_1}\},\mathtt{k}) \wedge} \\ (\mathbf{4}) & \multicolumn{4}{l}{1=1 \wedge 0=0 \wedge 1=1 \wedge 0=0 \wedge 0=0 \wedge} \\ (\mathbf{5}) & \multicolumn{4}{l}{1=1 \wedge 1=1} \end{array}$$

Now, most of the constraints are tautologies and we remove them. All that remains is a single constraint:

$$(\mathbf{3}) \quad \mathtt{sum\_leq}(\{\neg\mathtt{H}_{\mathtt{X}_1},\neg\mathtt{H}_{\mathtt{X}_1},\mathtt{H}_{\mathtt{X}_1}\},\mathtt{k})$$

which is satisfied for $k = 1$ when $\mathtt{H}_{\mathtt{X}_1}=1$, and as implied from θ then we have $\mathtt{H}_{\mathtt{A}_1}=1, \mathtt{H}_{\mathtt{A}_2}=1, \mathtt{H}_{\mathtt{X}_2}=1, \mathtt{H}_{\mathtt{O}_1}=0$. This example illustrates how equi-propagation and partial evaluation are applied to simplify constraints prior to their encoding to CNF.

We summarize with the following observations:

1. The constraint model for MBD is polynomial in the size of the system: each component contributes a constraint and a fresh health variable, each cone contributes an assignment to the health variables for its dominated components, each section contributes a cardinality constraint, and finally the observation contributes an assignment to the input and out variables of the system.

2. Constraint simplification using BEE is polynomial in the size of the constraint model. This is because: (a) each simplification step reduces the number of Boolean variables in the model by at least one — so there are a linear number of steps, and (b) each step checks the applicability of a fixed number of simplification patterns to each constraint.

3. The CNF encoding of the constraint model is polynomial in size, as each of the constraints introduces a polynomial number of clauses to the CNF (all of the constraints supported by BEE have this property).

6. Process and Implementation

In this section we summarize the different phases of the diagnosis process in our approach. In Section 6.1 we focus on the case where we seek a single minimal cardinality diagnosis, and then in Section 6.2, on the case when we seek all minimal cardinality diagnoses.

6.1 Single Minimal Cardinality Diagnosis

The process for a single minimal cardinality diagnosis consists of four phases. Let $\mu = \langle \mathit{SD}, \mathit{COMPS}, \mathit{OBS} \rangle$ be an MBD problem. In the first two phases we construct a constraint model. First, focusing on SD, to introduce constraints which are independent of the observation, and then "per observation" to introduce further constraints. In the third phase we encode the constraint model to a CNF, φ_μ^k where k is an upper bound on the size of the minimal cardinality diagnosis. In the fourth phase, solving φ_μ^k using a SAT solver results in a diagnosis with cardinality at most k. We now detail these four phases.

Phase 1. Modeling the system ("offline"): The system SD is first preprocessed to partition it into sections (Algorithm 1) and cones (Algorithm 2). Then, we introduce Constraints 1 to model SD in terms of its components behavior and introduce Constraints 4 to bound the number of unhealthy components per section. Finally, using information about cones, we add Constraint 3 which asserts that, without loss of generality, all dominated components are healthy. All of the system preprocessing is performed "offline", once per system.

Phase 2. Modeling the observation ("online"): Constraint 5 is added to model the observation OBS, and Constraint $2'$ is added to bound the total number of unhealthy components by the upper bound k_{UB} obtained by application of Algorithm 3.

Phase 3. Encoding: The constraint system is then simplified "online", for each observation and encoded to a CNF φ_μ^k, applying the optimizing CNF compiler (Metodi et al., 2011). The parameter k reflects the bound set in Constraint $2'$ to bound the number of unhealthy components in a diagnosis. Initially, k is computed by Algorithm 3.

Phase 4. Solving: To compute a diagnosis, Δ, we seek a satisfying assignment for the encoding, φ_μ^k, by applying the CryptoMiniSat solver (Soos, 2010). Δ is then the set of health variables assigned false by this assignment. Denoting $|\Delta| = k'$, we again seek a satisfying assignment, but this time for the formula $\varphi_\mu^{k'-1}$. If a satisfying assignment is found, it indicates a smaller diagnosis, Δ'. Otherwise, Δ is of minimal cardinality. This process is invoked repeatedly, each time finding a smaller diagnosis, until for some k' the formula $\varphi_\mu^{k'-1}$ is not satisfiable. Then, the diagnosis found in the previous iteration is of minimal cardinality.

To facilitate the search for a minimal cardinality diagnosis, we apply the SAT solver wrapper, SCryptoMiniSat (Metodi, 2012b). SCryptoMiniSat takes as input a CNF formula (φ_μ^k) and the Boolean variables representing the number k. It provides a satisfying assignment which minimizes k. SCryptoMiniSat takes advantage of the incrementality of the underlying SAT solver which maintains learned clauses over consecutive calls which only add clauses.

Algorithm 4 illustrates the process of finding a single minimal cardinality diagnosis (identified at line 17), and of returning the set of all minimal cardinality diagnoses (line 23).

6.2 All Minimal Cardinality Diagnoses

To find all minimal cardinality diagnoses we first apply the process described in Section 6.1 to find a single minimal cardinality diagnosis. This provides us with the value k' indicating the number of faulty components in a minimal cardinality diagnosis.

Then, to enumerate the set of all top-level minimal cardinality diagnoses (each of size k') we apply an additional functionality of SCryptoMiniSat which allows to enumerate (possibly with a

Algorithm 4 SATbD

input: A system *SD* with components *COMPS* and an observation *OBS*

output: Ω, the set of minimal cardinality diagnoses

```
1: // Phase 1: Offline pre-processing
2: Sections ← partition SD into sections                                   // Algorithm 1
3: Constraints ← $\emptyset$
4: for all $(S \in Sections)$ do
5:     for all $(c \in S)$ do
6:         Add Constraints 1 to Constraints                 // describes normal behavior of component $c$
7:         Dominators ← dominators$(c, S, SD)$                                  // Algorithm 2
8:         if $|Dominators| > 1$ then
9:             Add Constraint 3 for component $c$ to Constraints           // sets dominated gates to healthy
10: // Phase 2: Modeling the observation
11: Add Constraint 5 to Constraints                                  // add constraints representing OBS
12: $\Delta$ ← Find initial diagnosis                                         // Algorithm 3
13: Add Constraint $2'$ to Constraints with $\mathtt{k}_{\mathtt{UB}} = |\Delta|$
14: // Phase 3: Encoding
15: $\varphi \leftarrow$ BEE(Constraints)                     // run the constraint compiler to obtain a CNF
16: // Phase 4: Solving
17: $\Delta_{MC}$ ← SCryptoMiniSatMinimize($k$,$\varphi$)   // find min. card. diagnosis (assignment that minimizes $k = |\Delta|_{MC}$)
18: // Finding all diagnoses of minimal cardinality
19: $\varphi \leftarrow \varphi \wedge cnf(\mathtt{k}_{\mathtt{UB}} = |\Delta_{MC}|)$       // restrict Constraint $2'$ to use $\mathtt{k}_{\mathtt{UB}} = |\Delta_{MC}|$
20: $\Omega \leftarrow \emptyset$, $\Omega_{TLD}$ ← SCryptoMiniSatAllSolutions($\varphi$)          // find all top-level diagnoses
21: for all $(\omega \in \Omega_{TLD})$ do
22:     Add to $\Omega$ all the diagnoses expanded from $\omega$                      // Proposition 2
23: return $\Omega$
```

	System Details				Feldman				DXC-09				Sidd.
Name	|COMP|	**in**	**out**	**offline**	#obs.	1-10	11-20	21-30	#obs.	1-10	11-20	21-30	#obs.
74181	65	14	8	0.02	350	100	0	0	54	100	0	0	-
74182	19	9	5	0.01	250	100	0	0	50	100	0	0	-
74283	36	9	5	0.01	202	100	0	0	30	100	0	0	-
c432	160	36	7	0.03	301	100	0	0	45	100	0	0	700
c499	202	41	32	0.08	835	93	7	0	141	73	27	0	400
c880	383	60	26	0.06	1182	44	44	12	198	41	40	19	800
c1355	546	41	32	0.24	836	87	13	0	98	79	21	0	800
c1908	880	33	25	0.37	846	93	7	0	127	74	26	0	800
c2670	1193	233	140	0.29	1162	66	34	0	168	52	43	5	800
c3540	1669	50	22	0.71	756	98	2	0	36	100	0	0	-
c5315	2307	178	123	1.50	2038	47	45	8	248	43	40	17	40
c6288	2416	32	32	1.48	404	100	0	0	1	100	0	0	40
c7552	3512	207	108	1.73	1557	49	46	5	176	45	41	14	40

Table 1: The Benchmark suite: systems 74XXX and ISCAS-85, and observations: Feldman, DXC-09 and Siddiqi.

specified time-out) all, or a specified number of, satisfying assignments for a given CNF. We apply this option to enumerate all satisfying assignments for the formula, φ_{μ}^{k} described in Section 6.1 with $k = k'$.

Finally, based on Proposition 2 we expand the obtained top-level diagnoses to provide all minimal cardinality diagnoses. Note that the observation that a TLD can be expanded easily to all minimal cardinality diagnoses applies for any diagnosis algorithm. As such, diagnosis algorithms in general can focus, and be compared on finding TLDs, instead of finding all minimal cardinality diagnoses.

7. Experimental Results

This section presents an experimental evaluation of our proposed SAT-based encoding for MBD. In Section 7.1, we consider the search for a single minimal cardinality diagnosis and compare the performance of SATbD to the algorithms: HA* (Feldman & van Gemund, 2006), CDA* (Williams & Ragno, 2007) and SAFARI (Feldman et al., 2010a). In Section 7.2, we consider the search for all minimal cardinality diagnoses and compare SATbD to the algorithms: HDIAG (Siddiqi & Huang, 2007) and DCAS (Siddiqi & Huang, 2011). Finally, in Section 7.3, we evaluate the impact of the various components of our SAT-based encoding of MBD. All experiments were run on an Intel Core 2 Duo (E8400 3.00GHz CPU, 4GB memory) under Linux (Ubuntu lucid, kernel 2.6.32-24-generic) unless stated otherwise. The entire set of our tools, range of benchmarks, as well as a more detailed report of the results can be found online (Metodi, 2012a; Metodi, Stern, Kalech, & Codish, 2012b).

Table 1 provides basic details concerning the systems and three observation sets in our benchmark suite. The systems are 74XXX (Hansen, Yalcin, & Hayes, 1999), described in the first 3 rows, and ISCAS-85 (Brglez et al., 1989), described in the following 10 rows. The left column in the table specifies the system name. The next four columns (from the left) describe the systems: the numbers of components, inputs and outputs in each of the systems, and also the preprocessing time per system for the SAT-based approach. This includes all actions performed "once per system" as described in Section 6.1: decomposing the system to sections and cones and computing the bounds per section.

The rest of the columns are divided to three groups describing experiments with the three observation sets. The first two describe the observation set generated by Feldman et al. (2010a) and the DXC-09 observation set used in the diagnosis competition (DXC) of 2009. These are applied in our experimentation to evaluate the search for a single minimal cardinality diagnosis. The minimal cardinality of the diagnoses in these observations is between 1 and 30. The columns in each of these two groups indicate the number of observations and a distribution of the observations according to the size of their minimal cardinality diagnoses. The observations in these sets are considered hard and many of them, have high minimal cardinality diagnoses. The third group (the rightmost column in the table) presents the observation set generated by Siddiqi and Huang (2011) where minimal cardinality is bounded by 8 and the observations are distributed uniformly according to the size of their minimal cardinality diagnoses. This set is used in the evaluation of the search for all minimal cardinality diagnoses.

Table 1 illustrates a comprehensive experimental benchmark involving a total of 28,527 observations of varied minimal cardinality diagnosis size. Observe also that the (offline) preprocessing time per system is negligible. For instance, reprocessing the largest system, `c7552`, takes less than two seconds.

7.1 SATbD vs. Other MBD Algorithms: Single Minimal Cardinality Diagnosis

We now compare SATbD to the algorithms: HA* (Feldman & van Gemund, 2006), CDA* (Williams & Ragno, 2007) and SAFARI (Feldman et al., 2010a) in their application to search for a single minimal cardinality diagnosis. HA* and CDA* are based on a complete algorithm to find all minimal subset diagnoses — those that do not contain other diagnoses. They can be configured such that the first minimal subset diagnosis is guaranteed to be also of minimal cardinality and it is this configuration that we apply for comparison with SATbD. SAFARI applies an algorithm based on stochastic search which does not guarantee minimal cardinality and not even minimal subset diagnosis. Feldman et al. (2010a) report that even for single and double fault cardinalities, SAFARI does not always find the minimal cardinality. So, at the expense of minimality, SAFARI is often faster, comparing to HA* and CDA*.

	HA*		CDA*		SAFARI				SATbD	
Name	Succ. rate%	Time Sec.	Succ. rate%	Time Sec.	Succ. rate%	Min card.%	Diag. ratio	Time Sec.	Succ. rate%	Time Sec.
74181	68.3	3.15	46.3	4.51	100.0	44.0	1.33	0.00	100.0	0.02
74182	100.0	0.00	100.0	0.01	100.0	91.0	1.04	0.00	100.0	0.01
74283	100.0	0.04	100.0	1.45	100.0	57.0	1.28	0.00	100.0	0.02
c432	78.1	3.63	38.2	5.15	100.0	28.0	1.68	0.03	100.0	0.03
c499	24.1	5.45	10.1	1.22	100.0	7.0	2.00	0.05	100.0	0.04
c880	11.9	3.76	6.3	6.66	100.0	48.0	1.09	0.18	100.0	0.05
c1355	11.4	3.90	0.0	-	100.0	5.0	1.96	0.37	100.0	0.07
c1908	6.4	1.75	0.0	-	100.0	17.0	1.92	1.08	100.0	0.14
c2670	12.3	4.83	0.0	-	100.0	14.0	1.52	2.71	100.0	0.15
c3540	3.7	4.30	0.0	-	100.0	9.0	2.06	5.25	100.0	0.27
c5315	2.7	11.94	0.0	-	100.0	9.0	1.96	13.34	100.0	0.42
c6288	13.6	7.87	0.0	-	53.5	25.0	7.88	16.18	100.0	0.56
c7552	4.2	1.06	0.0	-	0.0	-	-	-	99.3	1.07
80 sec timeout										
c7552	7.3	20.77	0.0	0.0	99.5	13.0	1.68	43.50	100.0	1.49

Table 2: Single minimal cardinality diagnosis, Feldman's observations (30 sec. timeout).

Algorithm	HA*			CDA*			SAFARI			SATbD		
System	Min	Max	St. dev.	Min	Max	St. dev.	Min	Max	St. dev.	Min	Max	St. dev.
74181	0.00	29.40	6.04	0.00	29.01	7.30	0.00	0.01	0.00	0.00	0.03	0.01
74182	0.00	0.00	0.00	0.00	0.08	0.02	0.00	0.00	0.00	0.00	0.02	0.00
74283	0.00	0.76	0.08	0.00	28.14	2.79	0.00	0.00	0.00	0.00	0.03	0.01
c432	0.00	29.53	6.33	0.04	26.26	7.00	0.02	0.03	0.00	0.00	0.05	0.01
c499	0.02	21.43	5.03	0.88	1.55	0.21	0.04	0.06	0.00	0.01	0.07	0.01
c880	0.01	29.43	6.70	0.24	29.87	8.18	0.14	0.21	0.01	0.01	0.14	0.01
c1355	0.04	22.12	3.88	-	-	-	0.32	0.43	0.02	0.02	0.10	0.02
c1908	0.44	14.78	2.51	-	-	-	0.95	1.19	0.04	0.03	0.52	0.04
c2670	0.05	27.91	6.88	-	-	-	2.44	3.03	0.09	0.04	0.22	0.04
c3540	0.22	29.42	6.26	-	-	-	4.78	6.20	0.16	0.06	0.84	0.10
c5315	0.19	29.68	11.98	-	-	-	12.24	14.68	0.36	0.09	5.93	0.31
c6288	0.21	28.32	8.38	-	-	-	13.27	30.96	4.78	0.10	1.27	0.22
c7552	0.47	9.47	1.86	-	-	-	30.00	30.22	0.01	0.14	24.30	2.03

Table 3: Single minimal cardinality diagnosis, Feldman's observations, additional statistics.

Table 2 presents the evaluation focusing on Feldman's observations imposing a 30 second timeout (except for the bottom line). The columns indicate for each algorithm, the percentage of observations solved within the prescribed timeout (`Succ. rate %`) and the average search time (`Time Sec.`) where the average is computed over the set of observations excluding timeouts. For SATbD, the search time: (1) includes all actions performed "once per observation" as described in Section 6.1 (Modeling the observation ("online")); (2) Excludes the cost of actions performed once per system as described in Table 1 of Section 6.1 (column "offline"); and (3) includes times for adding the observation and the cardinality constraints, encoding to CNF and solving with SCryptoMiniSat.

The results in Table 2 show clearly that SATbD outperforms all of the other evaluated algorithms, both in terms of success rate as well as in terms of average runtime. SATbD also outperforms SAFARI which succeeds to compute a diagnosis for almost all of the systems. However, only a small percentage of these are of minimal cardinality as indicated by the column "`Min card%`" which shows the percentage (excluding timeouts) of observations where the diagnosis found by SAFARI is actually of minimal cardinality. We also show, in the column titled "`Diag. ratio`", the ratio between the average cardinality of the diagnoses found by SAFARI and the average minimal cardinality. So for example looking at the data for system `74181`, 44% of the diagnoses found by SAFARI are minimal, and the average diagnosis size found is 1.33 times larger than the average size of the minimal cardinality diagnosis. We observe that SATbD computes and verifies minimal cardinality diagnoses even for observations with a minimal cardinality of 30. To the best of our knowledge, no algorithm before succeeded to compute minimal cardinality diagnosis for such hard observations.

Table 3 details additional statistics for the running times presented in Table 2, showing the minimum, maximum, and standard deviation of the runtime for the solved observations. As can be seen, the standard deviation of SATbD is very small compared to the other algorithms. The displayed statistics are only over cases solved within the 30 second timeout. Table entries with "-" mark cases where the corresponding algorithms could not find a minimal cardinality diagnosis (within the timeout) even for a single observation.

Observe in Table 2 that 99.3% of the 1,557 observations for system `c7552` are solved by SATbD within the 30 second timeout (only 11 observations are not solved). All of the observations are, however, solved within 80 seconds each, as indicated by the last row in the table. One may observe that while SATbD solves all observations given 80 seconds, the other algorithms are still not able to solve all of them, with the exception of SAFARI. Given the extended 80 seconds timeout, SAFARI is also able to solve almost all of the observations for this system, but returns minimal cardinality diagnoses only in 13% of the cases, and in average, the size of the diagnosis found by SAFARI is 1.68 times larger than the actual minimal cardinality.

	HA*		CDA*		SAFARI			SATbD	
Name	Succ. rate%	Time Sec.	Succ. rate%	Time Sec.	Succ. rate%	Min card.%	Time Sec.	Succ. rate%	Time Sec.
74181	94.4	3.86	57.4	5.19	100.0	65	0.00	100.0	0.02
74182	100.0	0.00	100.0	0.01	100.0	90	0.00	100.0	0.00
74283	100.0	0.01	100.0	0.32	100.0	80	0.00	100.0	0.01
c432	75.6	3.70	40.0	7.17	100.0	22	0.03	100.0	0.03
c499	12.8	4.81	5.7	1.20	100.0	3	0.05	100.0	0.05
c880	10.1	5.51	4.5	9.66	100.0	48	0.18	100.0	0.05
c1355	9.2	3.33	0.0	-	100.0	4	0.38	100.0	0.07
c1908	7.1	3.10	0.0	-	100.0	14	1.12	100.0	0.20
c2670	11.3	5.86	0.0	-	100.0	18	2.85	100.0	0.14
c3540	11.1	1.23	0.0	-	100.0	31	5.83	100.0	0.29
c5315	1.2	19.77	0.0	-	100.0	7	13.12	100.0	0.62
c6288	100	0.24	0.0	-	100.0	100	25.28	100.0	0.1
c7552	2.3	0.47	0.0	-	0.0	-	-	100.0	1.01

Table 4: Single minimal cardinality diagnosis, DXC-09 observations (30 sec. timeout).

Table 4 shows the evaluation for the DXC-09 benchmark and is in the same format as Table 2 (except that we omit the details regarding the size of the diagnoses found by SAFARI). The results exhibit the same trend: our SAT-based method is substantially faster than all of the previous algorithms. Note that this benchmark even contains observations with minimal cardinality diagnoses of 31 (!), which were also solved by SATbD under the 30 seconds timeout. Table 5 provides additional statistics for these runtimes in the same format as in Table 3. Here too, we see a small standard deviation for SATbD compared to the other algorithms. Note that the data for the c6288 system is not given, as there is only a single observation for this system in the DXC-09 observation set.

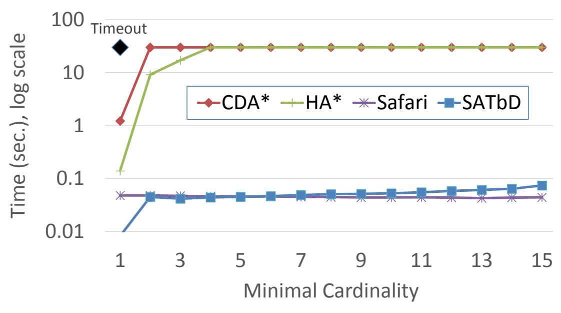
Figure 6 details the evaluation for a single minimal cardinality diagnosis with Feldman's observations for four systems (from smaller to larger). For each system we plot the average runtime to find a single minimal cardinality diagnosis (including timeouts) as a function of the value of the minimal cardinality. The black diamond labeled "Timeout" marks the time limit, after which algorithms were halted.

Typically, the diagnosis problem becomes harder as the minimal cardinality increases. First consider the three plots in Figures 6a, 6b and 6c. The two upper curves correspond to the systems HA* and CDA* and they quickly converge to the 30 seconds timeout. The curves for SAFARI are more or less constant but only a minority of the diagnoses are actually of minimal cardinality (7% in `c499`, 48% in `c880` and 9% in `c5315`). The performance of SAFARI is not affected by the cardinality since it first finds an arbitrary diagnosis and then proceeds stochastically to minimize the diagnosis using a pre-defined number of attempts. This process involves consistency checks which are affected only by the size of the system and not by the cardinality of the diagnosis.

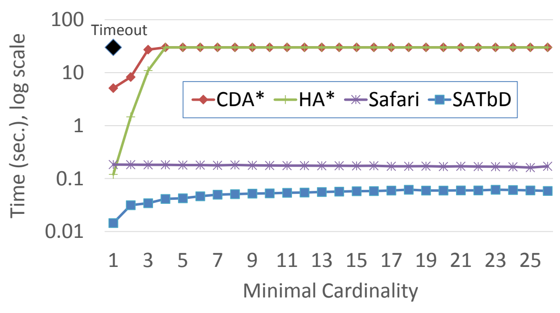
Now consider the plot in Figure 6d where we omit the curves for HA* and CDA* as they depict a constant 80 second timeout. The performance of SAFARI is more or less constant (around the average 43.5 seconds) but only 13% of the diagnoses found are actually of minimal cardinality. In contrast, SATbD is considerably faster and scales to solve even the hardest observations.

Algorithm	HA*			CDA*			SAFARI			SATbD		
System	Min	Max	St. dev.	Min	Max	St. dev.	Min	Max	St. dev.	Min	Max	St. dev.
System	Min	Max	St. dev.	Min	Max	St. dev.	Min	Max	St. dev.	Min	Max	St. dev.
74181	0.00	27.26	6.70	0.00	20.12	7.44	0.00	0.01	0.00	0.00	0.03	0.01
74182	0.00	0.00	0.00	0.00	0.08	0.02	0.00	0.00	0.00	0.00	0.02	0.00
74283	0.00	0.02	0.01	0.01	1.22	0.40	0.00	0.00	0.00	0.00	0.03	0.01
c432	0.00	23.06	5.98	0.05	26.14	8.75	0.03	0.04	0.00	0.01	0.04	0.01
c499	0.01	11.34	4.99	0.93	1.42	0.16	0.04	0.07	0.01	0.01	0.07	0.01
c880	0.01	29.60	8.34	0.27	20.33	8.97	0.16	0.24	0.01	0.02	0.07	0.01
c1355	2.74	6.52	1.20	-	-	-	0.34	0.42	0.02	0.02	0.09	0.02
c1908	0.85	15.34	4.63	-	-	-	1.00	1.25	0.05	0.03	1.36	0.16
c2670	0.05	26.04	8.06	-	-	-	2.56	3.07	0.11	0.05	0.21	0.03
c3540	0.31	3.44	1.49	-	-	-	5.53	6.18	0.16	0.06	0.84	0.20
c5315	9.86	28.39	9.34	-	-	-	12.10	14.11	0.32	0.10	10.06	0.90
c7552	0.47	0.47	0.00	-	-	-	-	-	-	0.17	11.54	1.54

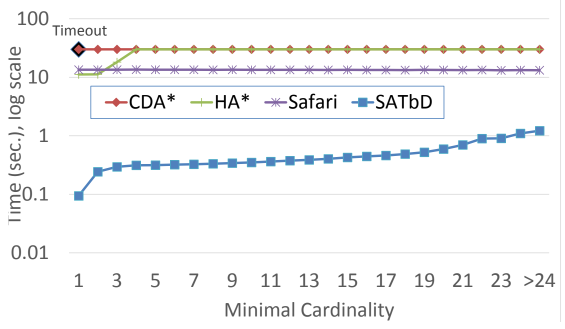
Table 5: Single minimal cardinality diagnosis, DXC-09 observations (30 sec. timeout), additional statistics.



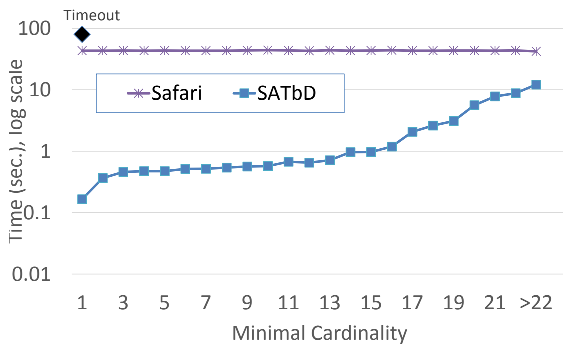
(a) System `c499` (30 sec. timeout).



(b) System `c880` (30 sec. timeout).



(c) System `c5315` (30 sec. timeout).



(d) System `c7552` (80 sec. timeout).

Figure 6: Single minimal cardinality diagnosis, Feldman's observations, average search time per size of the minimal cardinality diagnosis.

The results of this section clearly indicate that our SAT based approach outperforms the other three algorithms in the search for a single minimal cardinality diagnosis.

	IntelXeon X3220, 2.4GHz, 2Gb RAM				Intel Core 2-Duo E8400, 3.00GHz, 4GB RAM				
	HDIAG		DCAS		SATbD				
Name	Succ. rate%	Time Sec.	Succ. rate%	Time Sec.	Succ. rate%	TLD Sec.	Time Sec.	TLD count	ALL count
c432	100.0	0.21	100.0	0.31	100.0	0.07	0.09	7.9	72
c499	100.0	0.12	100.0	0.20	100.0	0.08	0.10	2.4	345
c880	99.0	0.07	99.0	0.12	100.0	0.08	0.11	10.3	963342
c1355	99.5	0.16	99.5	0.15	100.0	0.13	0.16	2.8	331927
c1908	90.5	368.13	76.5	82.25	100.0	0.25	0.30	19.9	1894733
c2670	90.0	176.17	100.0	3.15	100.0	0.23	0.29	6.7	8492
c5315	0.0	-	97.5	52.34	100.0	0.58	0.67	26.4	356949
c6288	0.0	-	27.5	305.10	50.0	104.58	105.14	7231.8	37499
c7552	0.0	-	87.5	260.93	100.0	1.01	1.12	64.9	68396

Table 6: Siddiqi's observation set: search for all minimal cardinality diagnoses (1800 sec. timeout).

7.2 SATbD vs. Other MBD Algorithms: All Minimal Cardinality Diagnosis

We now compare SATbD to the algorithms HDIAG (Siddiqi & Huang, 2007) and DCAS (Siddiqi & Huang, 2011) in their application to search for all minimal cardinality diagnosis. Both algorithms search for a complete set of the minimal cardinality diagnoses. We consider the observations generated by Siddiqi and Huang (2011) where minimal cardinality is bounded by 8.

Table 6 presents the results of our evaluation. The results for the HDIAG and DCAS are quoted from Siddiqi and Huang's work (2011) where experiments are reported for an IntelXeon X3220 2.4GHz, 2Gb RAM. We present results only for the systems for which Siddiqi and Huang reported results. Although the machines differ (with an advantage for SATbD), the results show a clear advantage for SATbD which is faster in orders of magnitude for the larger systems.
For each of the three algorithms the table reports on:

Succ. rate% indicating the percentage of the observations for which the algorithm finds all minimal cardinality diagnoses within an 1800 second timeout; and

Time sec. indicating the average computation times to find all minimal cardinality diagnoses (taking the average over the set of observations for which there is no time out).

For SATbD we report also

TLD sec. the average runtime to compute all top level diagnoses;

TLD count the number of top level diagnoses; and

ALL count the total number of minimal cardinality diagnoses found.

Table 6 illustrates that SATbD clearly outperforms HDIAG and DCAS. It succeeds to compute all minimal cardinality diagnoses for all observations for all of the systems except for `c6288` where it succeeds on 50% of the 40 observations compared to 26.5% for DCAS. Note that because of the higher success rate, the average runtimes for SATbD involve harder observations not solved by DCAS.

Observe that most of the diagnosis time for SATbD is spent to find all top level diagnoses indicated in column `TLD sec`. The cost to compute all minimal cardinality diagnoses is indicated in column `Time sec` and the difference between these two columns is negligible. This reflects the fact that the set of all minimal cardinality diagnoses is derived as a cross product representation of the set of all minimal cardinality diagnoses. Observe also that the number of TLDs (column `TLD count`) is small in comparison to the huge number of minimal cardinality diagnoses (column `ALL count`). The focus on TLDs is essential as each additional solution invokes an additional call to the SAT solver. Using the SAT solver to find the minimal cardinality diagnoses directly would be hopeless due to their sheer number.

7.3 Impact of the Components of SATbD

We proceed to illustrate the impact of the various components of our SAT-based encoding of MBD. SATbD is designed using a variety of techniques that distinguish it from the simple "vanilla" encoding of an MBD problem to a SAT problem described in Section 5.1. We present here an evaluation of the impact of these techniques based on several experiments using the following five configurations of our SAT based system. These configurations are "incremental": starting from the basic model, each one adds another component, ending with the final model applied in SATbD.

1. **Vanilla.** This is the minimal basic SAT encoding for MBD described in Section 5.1. Here we assume the naive upper bound on the minimal cardinality determined as the number of outputs of the given system.

2. **Improved Cardinality Bound.** Here we assume the same vanilla setting but consider the improved bound on the minimal cardinality of a diagnosis using Algorithm 3.

3. **E.P.** This setting is the same as the previous but applies the Equi-Propagation constraint compiler of Metodi and Codish (2011, 2012) to optimize the encoding as described in Section 5.5.

4. **Cones.** This setting is the same as the previous but also partitions the system into its cones and adds constraints to restrict the search to find top-level diagnoses (TLDs), as described in Section 3.

5. **Sections.** This setting is the same as the previous but also partitions the system into its sections and introduces corresponding redundant cardinality constraints as described in Section 5.3.

Figure 7 illustrates the impact of each of the five settings on the search for a single minimal cardinality diagnosis for 1182 observations from Feldman's observation set for system `c880`. On the horizontal axis, we consider the observations according to the size of their minimal cardinality diagnosis. On the vertical axis, we illustrate the average runtime in seconds. All runs apply a 300 second timeout. We choose system `c880` to present these results as: (a) it is the only midsize system for which the observation sets contain observations with minimal cardinality diagnosis larger than 20; and (b) it is the largest system which exhibits an interesting behavior for all five configurations without too many timeouts which shadow the results. With larger systems, such as `c3515` and `c7552` considered below, only the last two of the five configurations exhibit interesting curves. The other three configurations timeout on most of the observations.

The upper curve in Figure 7 describes the Vanilla setting and one may observe that as the cardinality increases the curve converges to the 300 second timeout. The second curve down describes the Improved Bound setting and illustrates the impact of Algorithm 3, especially for the observations with minimal cardinality diagnoses of size 1 – 10. Here, using the improved bound reduces

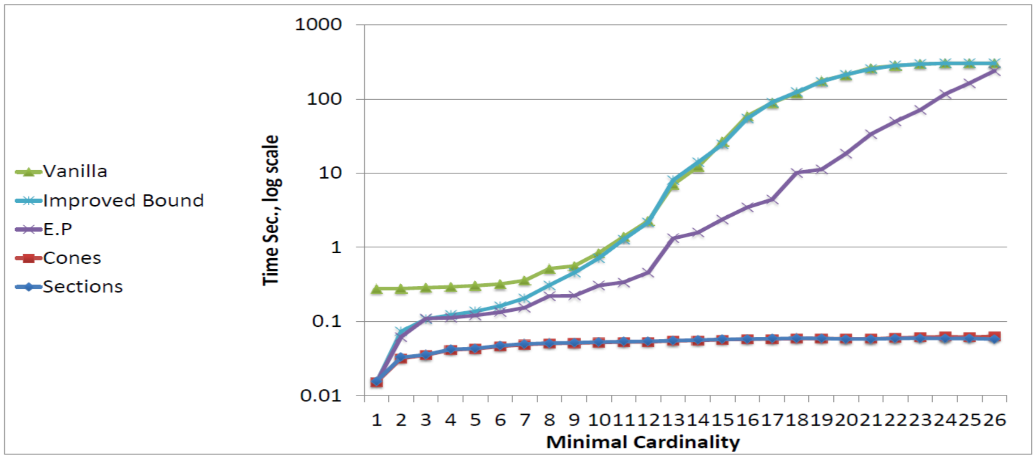
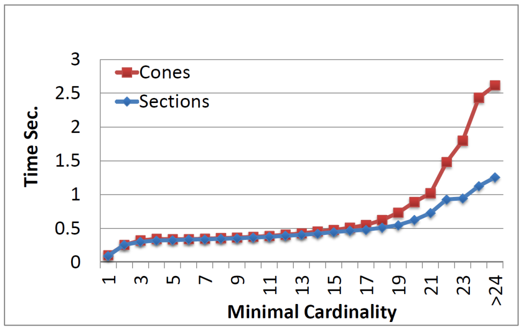


Figure 7: Impact of the different settings on the search for a single minimal cardinality diagnosis for system `c880`.

the number of iterations with the SAT solver and this is the source of the improvement. To explain why we see no improvement when the minimal cardinality diagnosis involves a larger number of faults, consider first that for large minimal cardinalities, the results are meaningless as both of the upper curves converge on the timeout. As for the medium sized minimal cardinalities, consider that with both techniques the overwhelming part of the runtime is spent on the last "UNSAT" iteration. Moreover, the number of iterations with the SAT solver with both techniques is more or less the same. This is because system `c880` has only 26 outputs which is the naive bound used in the vanilla setting which then jumps down in each iteration to a smaller bound (but not using a "one-by-one" decrement). Given this evaluation we might consider omitting the constraints for the improved lower bound depending on the parameters of the instance. However, the cost of running Algorithm 3 is negligible so we always impose the corresponding constraints.

The third curve down (E.P.) shows the additional impact when applying the Equi-Propagation constraint compiler. This results in substantial speedups over the first two settings. For example, finding a diagnosis of minimal cardinality 20 requires an average of 18.5 seconds with this setting in comparison to 214.9 seconds without it. The two lower curves of the graph coincide, and it is the fourth setting (cones) which makes the dramatic impact on performance for system `c880`. The average runtime required to find the first minimal cardinality using this setting is under 0.1 seconds for all observations. This includes finding diagnoses with minimal cardinality of 26 (!) in 58 milliseconds, on average. The runtimes with this setting are so small that we cannot observe any additional added value when applying the fifth setting (sections). To this end we consider in the next experiment a comparison using two larger systems, namely `c5315` and `c7552`.

Figure 8 illustrates the impact of the fifth setting which involves the partitioning of a system into sections. The left graph illustrates the impact of sections when seeking a single minimal cardinality diagnosis for the observations in system `c5315` and the right graph illustrates the impact when seeking all top-level minimal cardinality diagnoses (TLDs). The lower curve (in both graphs), summarizes the results using sections and the upper curve without. For example, for the observations with minimal cardinality 24, with sections finding the first minimal cardinality diagnosis requires



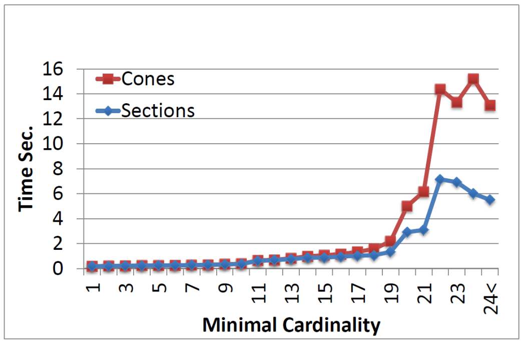
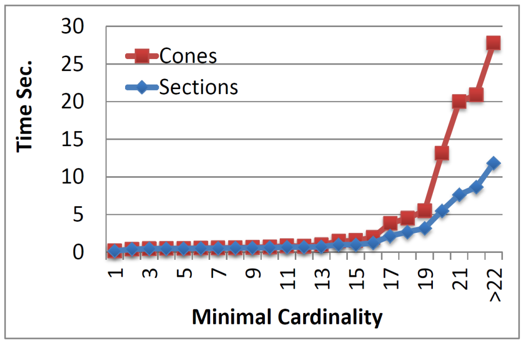


Figure 8: Impact of *sections* and *cones* on the search for a single minimal cardinality diagnosis (left) and on the search for all TLDs (right) using system `c5315`.



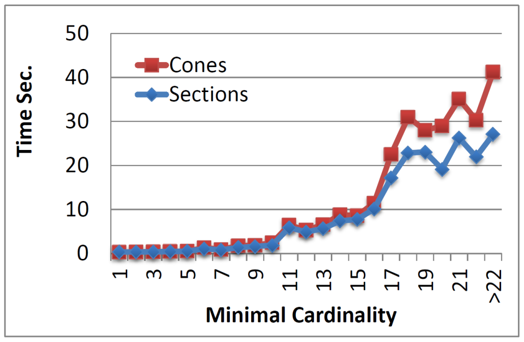


Figure 9: Impact of *sections* and *cones* on the search for a single minimal cardinality diagnosis (left) and on the search for all TLDs (right) using system `c7552`.

an average of 1.13 seconds, while without it requires 2.43 seconds. For the same observations, with sections, we find all TLDs in an average of 6 seconds, and without it requires 15 seconds.

Similar trends are illustrated in Figure 9 which depicts the same information for system `c7552`. Here we apply a timeout of 300 seconds, and this timeout is encountered for 48 of the 1,557 observations when searching for all minimal cardinality TLDs (timeout observations are not considered in the average runtimes). However, even for these 48 observations, using sections provides on average 50% more TLDs than without (15,785 vs. 9,585 TLDs) before reaching the timeout.

Table 7 details, for the ISCAS-85 benchmark with Feldman's observations, the average sizes of the SAT encodings when using the full SATbD algorithm. The table indicates the number of

System	c432	c499	c880	c1908	c2670	c3540	c5315	c6288	c7552
# Components	160	202	383	880	1193	1669	2307	2416	3512
# Variables	190	240	227	1026	636	2051	2407	7161	3525
# Clauses	556	1193	758	4063	2055	7456	11277	22061	12731

Table 7: Average SAT encoding sizes for the ISCAS-85 benchmark with Feldman's observations.

components in each system and the number of variables and clauses in the resulting CNFs, taking the average over all observations in the Feldman benchmark. Notably, as indicated by the table, the SAT encodings are extremely small with (in the worst case) less than three CNF variables and ten CNF clauses per system component.

8. Discussion

This paper focuses on an MBD problem considering possibly multiple faulty components in a weak fault model. The presentation is restricted to assume that: every component has a single output, the observation includes a single input/output, there are no queries about observations of specific components (probes), and there is no information regarding the probability of component failures. Even with these restrictions, the problem addressed in this paper is computationally hard and has been the focus of many prior works in the model-based diagnosis literature (Feldman & van Gemund, 2006; Williams & Ragno, 2007; Feldman et al., 2010a; Siddiqi & Huang, 2007, 2011).

We discuss here briefly the applicability of our approach in a more general setting. Deeper analysis of how to adapt our approach to a more general setting is a research topic on its own. We believe that our success in applying SAT solvers to MBD problems in this simplified setting paves the way to their application in other more general settings.

8.1 Boolean Extensions

Our approach applies directly in any setting where the components are described by propositional formulae and where their mode of fault can be ignored (i.e., any weak fault model setting). Boolean circuits are just one straightforward example where this is obvious and where the community has focused attention. Other examples with similar assumptions and where our SAT-based approach can be expected to apply directly include the works by: Abreu et al. (2009) where the authors model software components as propositional formulae and apply an MBD algorithm to find bugs; Kalech and Kaminka (2005, 2006) where the authors model robots in a multi-robot system and diagnose the violation of coordination constraints among robots; and Felfernig et al. (2012) where the authors model finite domain constraints and diagnose inconsistent constraint sets.

Note that restricting components to have a single output is not really a restriction as it is straightforward to represent a component with multiple outputs as a conjunction of single output components. In this way the partition into cones and sections is fully compatible with multiple-output components.

8.2 Probabilities

Real world applications of MBD typically come with information regarding the probability of a component to be faulty and many MBD algorithms exploit this information to prioritize diagnoses with respect to their likelihood (interalia, see de Kleer & Williams, 1989; Williams & Ragno, 2007). Sachenbacher and Williams (2004) showed how to incorporate fault probabilities in a tree-decomposition diagnosis algorithm. Extending our approach to consider probabilities is straightforward. The essential difference is in Constraint 2, $\texttt{sum\_leq}(\left\{ \neg \texttt{H}_\texttt{c} \,\middle|\, \texttt{c} \in \mathit{COMPS} \right\}, \texttt{k})$, which specifies the "objective function" we aim to minimize. It can be replaced with a constraint that takes probabilities into account: $\texttt{sum\_leq}(\left\{ \neg \texttt{H}_\texttt{c} \cdot (1 - \texttt{p}_\texttt{c}) \,\middle|\, \texttt{c} \in \mathit{COMPS} \right\}, \texttt{k})$ where p_c is the probability that component c is faulty. So, constraints which are more likely to be faulty contribute less to the objective function. Note that it is straightforward to normalize the constraint so that the coefficients are integers. Constraints of this form are called Pseudo-Boolean constraints and their encoding to CNF is well studied (Eén & Sörensson, 2006).

8.3 Testing and Probes

Another extension that is straightforward to model in SAT concerns testing and probing (de Kleer & Williams, 1987). Here, the diagnosis algorithm is given multiple observations on the input/output behavior of the system (in testing) or additional observations on internal "wires" in the system (in probing). Under the assumption that faulty components are consistently faulty, we can invalidate diagnoses that are inconsistent with multiple observations. Similarly, probes can invalidate diagnoses that are not consistent with the new internal observation. Both methods can be run iteratively until there is a single consistent diagnosis. Both techniques are straightforward to encode to SAT. For testing, to improve the diagnosis we simply take the conjunction of encodings with respect to different observations, but using the same health variables. For probing we also take a conjunction with the internal observations. The main challenge for both methods is to reduce the number of probes (or tests) required to find the actual diagnosis. A common, greedy, approach to address this challenge is to choose a probe (test) that maximizes the information gain as described by Feldman et al. (2010b).

8.4 The Strong Fault Model

Now consider an extension of our approach to a setting in which components are associated with a wider range of possible faulty behavior modes. This is called the "Strong Fault Model". For instance, a setting where a circuit component may be "stuck at 0" (always returns output 0), "stuck at 1"(always returns output 1), or "flip" (always flips its output) (Struss & Dressier, 1989; de Kleer & Williams, 1989). In the context of this example, a naive SAT model may be obtained as follows. Instead of considering only a single propositional health variable $\mathtt{H_c}$, consider one additional variable per fault mode: $\mathtt{H_c^{s0}}$ (stuck at zero), $\mathtt{H_c^{s1}}$ (stuck at one), and $\mathtt{H_c^{f}}$ (flip). Now, introduce clauses (a) to express that any fault mode is a fault ($\mathtt{H_c} \leftrightarrow \mathtt{H_c^{s0}} \wedge \mathtt{H_c^{s1}} \wedge \mathtt{H_c^{f}}$); and (b) to express that there is at most one fault on a component $\mathtt{sum\_leq}(\{\neg\mathtt{H_c^{s0}}, \neg\mathtt{H_c^{s1}}, \neg\mathtt{H_c^{f}}\}, \mathtt{1})$. In this way the propositional health variable H_c, as before, indicates if a component is healthy but a diagnosis is now an assignment of fault types to each component. However, this extension requires to reconsider the definitions of minimal diagnosis and cardinality and presents a new challenge in identifying useful partitions of the system and in solving the problem with SAT. We consider this as future work.

8.5 Diagnosis for Physical Systems

This is the most challenging problem. Physical systems are typically dynamic, involve components with time dependent behavior, and described in terms of continuous variables. One common approach to apply MBD to physical systems is to use qualitative models where the behavior of the system is modeled as a set of constraints over non-numerical (discrete) descriptions (Subramanian & Mooney, 1996). A well-known example of an MBD engine that makes such relaxations is the Livingstone Model-Based Diagnosis System (Williams & Nayak, 1996). Livingstone has been successfully applied in "Deep Space One", the first spacecraft for NASA's New Millennium program.

A first and important step for a SAT based approach for the diagnosis of physical systems is to provide a setting for MBD in the strong fault model and able to capture probabilities. Of course, there are many additional challenges and the modeling of such systems using SAT is an important but feasible challenge.

9. Conclusion

This paper addresses an MBD challenge which has been extensively researched for more than 25 years and for which a wide range of papers propose different algorithms. We present a novel SAT-based solution for this problem and determine for the first time, minimal cardinality diagnoses for the entire standard benchmarks. We present an extensive experimental evaluation comparing our algorithm to HA*, CDA*, SAFARI, HDIAG and DCAS. Results are unequivocal. Our algorithm outperforms the others, often by orders of magnitude, both for the search of a single minimal cardinality diagnosis as well as for the search for all minimal cardinality diagnoses. We succeed to find and verify a minimal cardinality diagnosis for all but 11 of the 28,257 observations of the benchmark in under 30 seconds per observation, and for the remaining 11 in under 80 seconds each. To the best of our knowledge, our SATbD algorithm is the first algorithm to find the minimal cardinality of these standard benchmarks discussed above. Further details regarding the experimental evaluation as well as a prototype implementation of our SAT-based MBD tool can be found online (Metodi, 2012a; Metodi et al., 2012b).

A major contribution to the success of our approach is the range of preprocessing techniques presented in Section 5. Their impact is demonstrated through five configurations of the system in Section 7.3 and their full combination is the fifth such configuration. Even as SAT, and other related solvers, improve we conjecture that careful modeling choices involving combinations of these techniques are invaluable to the success of future MBD algorithms. It is our belief that the results of this paper will pave the way to develop and apply SAT-based methodologies to other MBD problems. In particular, extensions for diagnosis with probabilities of components to be faulty, for sequential diagnosis with testing and probes, and, most challenging, for the diagnosis of physical systems with qualitative models. We expect that our methodology, which combines domain dependent preprocessing, clever modeling in SAT, and application of tools to optimize the CNF encodings, is relevant to other hard problems in AI where SAT-based techniques are applicable.

Acknowledgments

This research was supported by the Israel Science Foundation, grant 182/13.

References

Abreu, R., Zoeteweij, P., Golsteijn, R., & van Gemund, A. J. C. (2009). A practical evaluation of spectrum-based fault localization. *Journal of Systems and Software*, *82*(11), 1780–1792.

Asín, R., Nieuwenhuis, R., Oliveras, A., & Rodríguez-Carbonell, E. (2009). Cardinality networks and their applications. In Kullmann, O. (Ed.), *SAT*, Vol. 5584 of *Lecture Notes in Computer Science*, pp. 167–180. Springer.

Asín, R., Nieuwenhuis, R., Oliveras, A., & Rodríguez-Carbonell, E. (2011). Cardinality networks: a theoretical and empirical study. *Constraints*, *16*(2), 195–221.

Balakrishnan, K., & Honavar, V. (1998). Intelligent diagnosis systems. *Journal of Intelligent Systems*, *8*(3/4), 239–290.

Batcher, K. E. (1968). Sorting networks and their applications. In *AFIPS Spring Joint Computing Conference*, Vol. 32 of *AFIPS Conference Proceedings*, pp. 307–314.

Bauer, A. (2005). Simplifying diagnosis using LSAT: a propositional approach to reasoning from first principles. In Barták, R., & Milano, M. (Eds.), *International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR)*, Vol. 3524 of *Lecture Notes in Computer Science*, pp. 49–63, Berlin, Heidelberg. Springer-Verlag.

Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.). (2009). *Handbook of Satisfiability*, Vol. 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Brglez, F., Bryan, D., & Kozminski, K. (1989). Combinatorial profiles of sequential benchmark circuits. In *IEEE International Symposium on Circuits and Systems*, pp. 1929–1934.

Bylander, T., Allemang, D., Tanner, M. C., & Josephson, J. R. (1991). The computational complexity of abduction. *Artificial Intelligence*, *49*(1-3), 25–60.

Codish, M., & Zazon-Ivry, M. (2010). Pairwise cardinality networks. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, pp. 154–172.

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

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

de Kleer, J. (2008). An improved approach for generating max-fault min-cardinality diagnoses. In *International Workshop on Principles of Diagnosis (DX)*.

de Kleer, J., & Williams, B. C. (1989). Diagnosis with behavioral modes. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1324–1330.

Dressler, O., & Struss, P. (1995). Occ'm. `http://www.occm.de`.

DXC (2009). International diagnostic competition series. Website. https://sites.google.com/site/dxcompetition/.

Eén, N., & Sörensson, N. (2006). Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability (JSAT)*, *2*(1-4), 1–26.

El Fattah, Y., & Dechter, R. (1995). Diagnosing tree-decomposable circuits. *International Joint Conference on Artificial Intelligence (IJCAI)*, *95*, 1742–1749.

Feldman, A., Provan, G., de Kleer, J., Robert, S., & van Gemund, A. (2010). Solving model-based diagnosis problems with Max-SAT solvers and vice versa. In *International Workshop on Principles of Diagnosis (DX)*, pp. 185–192.

Feldman, A. (2012). Lydia-ng. `http://www.general-diagnostics.com/products.php`.

Feldman, A., de Castro, H. V., van Gemund, A., & Provan, G. (2013). Model-based diagnostic decision-support system for satellites. In *IEEE Aerospace Conference*, pp. 1–14. IEEE.

Feldman, A., Provan, G., & van Gemund, A. (2010a). Approximate model-based diagnosis using greedy stochastic search. *Journal of Artificial Intelligence Research (JAIR)*, *38*, 371–413.

Feldman, A., Provan, G., & van Gemund, A. (2010b). A model-based active testing approach to sequential diagnosis. *Journal of Artificial Intelligence Research (JAIR)*, *39*, 301–334.

Feldman, A., & van Gemund, A. J. C. (2006). A two-step hierarchical algorithm for model-based diagnosis. In *Conference on Artificial Intelligence (AAAI)*, pp. 827–833.

Felfernig, A., Schubert, M., & Zehentner, C. (2012). An efficient diagnosis algorithm for inconsistent constraint sets. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *26*(1), 53–62.

Fröhlich, P., & Nejdl, W. (1997). A static model-based engine for model-based reasoning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 466–473.

Fujiwara, H., Member, S., Shimono, T., & Member, S. (1983). On the acceleration of test generation algorithms. *IEEE Transactions on Computers*, *32*, 1137–1144.

Hansen, M. C., Yalcin, H., & Hayes, J. P. (1999). Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering. *IEEE Des. Test*, *16*, 72–80.

Jannach, D., & Schmitz, T. (2014). Model-based diagnosis of spreadsheet programs: a constraint-based debugging approach. *Automated Software Engineering*, *1*, 1–40.

Kalech, M., & Kaminka, G. A. (2005). Towards model-based diagnosis of coordination failures. In *Conference on Artificial Intelligence (AAAI)*, pp. 102–107.

Kalech, M., Kaminka, G. A., Meisels, A., & Elmaliach, Y. (2006). Diagnosis of multi-robot coordination failures using distributed CSP algorithms. In *Conference on Artificial Intelligence (AAAI)*, pp. 970–975.

Kirkland, T., & Mercer, M. R. (1987). A topological search algorithm for ATPG. In *ACM/IEEE Design Automation Conference*, DAC, pp. 502–508.

Knuth, D. E. (2014). *The Art of Computer Programming: Volume 4B, Pre-fascicle 6A, Section 7.2.2.2: Satisfiability*. Unpublished. Draft available from: `http://www-cs-faculty.stanford.edu/~knuth/fasc6a.ps.gz`.

Marques-Silva, J., Lynce, I., & Malik, S. (2009). Conflict-driven clause learning SAT solvers. *Handbook of satisfiability*, *185*, 131–153.

Metodi, A. (2012a). SCryptodiagnoser: A SAT based MBD solver. `http://amit.metodi.me/research/mbdsolver`.

Metodi, A. (2012b). SCryptominisat. `http://amit.metodi.me/research/scrypto`.

Metodi, A., & Codish, M. (2012). Compiling finite domain constraints to SAT with BEE. *Theory and Practice of Logic Programming (TPLP)*, *12*(4-5), 465–483.

Metodi, A., Codish, M., Lagoon, V., & Stuckey, P. J. (2011). Boolean equi-propagation for optimized SAT encoding. In *CP*, pp. 621–636.

Metodi, A., Codish, M., & Stuckey, P. J. (2013). Boolean equi-propagation for concise and efficient SAT encodings of combinatorial problems. *Journal of Artificial Intelligence Research (JAIR)*, *46*, 303–341.

Metodi, A., Stern, R., Kalech, M., & Codish, M. (2012a). Compiling model-based diagnosis to Boolean satisfaction. In *Conference on Artificial Intelligence (AAAI)*.

Metodi, A., Stern, R., Kalech, M., & Codish, M. (2012b). Compiling model-based diagnosis to Boolean satisfaction: Detailed experimental results and prototype implementation. `http://www.cs.bgu.ac.il/~mcodish/Papers/Pages/aaai-2012.html`.

Murray, J., Hughes, G., & Kreutz-Delgado, K. (2006). Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning Research (JMLR)*, *6*(1), 783.

Nica, I., Pill, I., Quaritsch, T., & Wotawa, F. (2013). The route to success - a performance comparison of diagnosis algorithms. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1039–1045.

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

Sachenbacher, M., & Williams, B. (2004). Diagnosis as semiring-based constraint optimization. In *Eureopean Conference on Artificial Intelligence (ECAI)*, pp. 873–877.

Selman, B., & Levesque, H. J. (1990). Abductive and default reasoning: A computational core. In *National Conference on Artificial Intelligence (AAAI)*, pp. 343–348.

Siddiqi, S. A., & Huang, J. (2007). Hierarchical diagnosis of multiple faults. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 581–586.

Siddiqi, S. A., & Huang, J. (2011). Sequential diagnosis by abstraction. *Journal of Artificial Intelligence Research (JAIR)*, *41*, 329–365.

Smith, A., Veneris, A. G., Ali, M. F., & Viglas, A. (2005). Fault diagnosis and logic debugging using Boolean satisfiability. *IEEE Trans. on CAD of Integrated Circuits and Systems*, *24*(10), 1606–1621.

Soos, M. (2010). Cryptominisat, v2.5.1. `http://www.msoos.org/cryptominisat2`.

Stein, B., Niggemann, O., & Lettmann, T. (2006). Speeding up model-based diagnosis by a heuristic approach to solving SAT. In *IASTED international conference on Artificial intelligence and applications*, pp. 273–278.

Stern, R. T., Kalech, M., Feldman, A., & Provan, G. M. (2012). Exploring the duality in conflict-directed model-based diagnosis. In *AAAI*.

Struss, P., & Dressier, O. (1989). "Physical negation": Integrating fault models into the general diagnostic engine. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1318–1323.

Struss, P., & Price, C. (2003). Model-based systems in the automotive industry. *AI magazine*, *24*(4), 17–34.

Stumptner, M., & Wotawa, F. (2001). Diagnosing tree-structured systems. *Artificial Intelligence*, *127*(1), 1–29.

Stumptner, M., & Wotawa, F. (2003). Coupling CSP decomposition methods and diagnosis algorithms for tree-structured systems. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 388–393.

Subramanian, S., & Mooney, R. J. (1996). Qualitative multiple-fault diagnosis of continuous dynamic systems using behavioral modes. In *National Conference on Artificial Intelligence (AAAI)*, pp. 965–970.

Torasso, P., & Torta, G. (2006). Model-based diagnosis through OBDD compilation: A complexity analysis. In *Reasoning, Action and Interaction in AI Theories and Systems*, pp. 287–305.

Wang, J., & Provan, G. (2010). A benchmark diagnostic model generation system. *Part A: Systems and Humans, IEEE Transactions on Systems, Man and Cybernetics*, *40*(5), 959–981.

Williams, B. C., & Nayak, P. P. (1996). A model-based approach to reactive self-configuring systems. In *National Conference on Artificial Intelligence (AAAI)*, pp. 971–978.

Williams, B. C., & Ragno, R. J. (2007). Conflict-directed A* and its role in model-based embedded systems. *Discrete Applied Mathematics*, *155*(12), 1562–1595.