A Multi-Valued SAT-Based Algorithm for Faster Model-Based Diagnosis

Alexander Feldman, **Jurryt Pietersma** and **Arjan van Gemund**
Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Mekelweg 4, 2628 CD, Delft, The Netherlands
Tel.: +31 15 2781935, Fax: +31 15 2786632
e-mail: {a.b.feldman,j.pietersma,a.j.c.vangemund}@tudelft.nl

Abstract

Finite integer domains offer an intuitive representation of fault diagnosis models of real-world systems. Approaches that encode multi-valued models to the Boolean domain suffer from combinatorial explosion. Prompted by recent advances in multi-valued SAT solving, in this paper we present a multi-valued diagnosis algorithm. This sound and complete algorithm is based on multi-valued SAT and A*, and does not require Boolean encoding. The resulting diagnostic engine is specifically designed to suit the characteristics of the diagnosis search and better exploits the locality which is present in the multi-valued variable domains of a wide-range of model-based diagnosis problems. Results from experiments on both synthetic and real-world problems are in agreement with recently reported good performance of multi-valued DPLL consistency checkers. Models used for experimentation include NASA's X-34 propulsion system and ASML's wafer scanner subsystems. The empirical results show that, depending on the domain size and number of variables, the multi-valued approach can deliver up to two orders of magnitude speedup over Boolean approaches.

1 Introduction

When considering a multi-valued model of the X-34 propulsion system [Sgarlata and Winters, 1997], one option is to model it in the Boolean domain and to use a SAT based diagnosis algorithm or, alternatively, to create a model, for which each variable is in the finite-integer domain. The latter approach allows for intuitive modeling of components with multiple fault modes and discretization of continuous values in areas like qualitative reasoning. Within this modeling technique, we face the choice of using directly a multi-valued diagnostic algorithm or, as an alternative, to encode the model in the Boolean domain by using an appropriate mapping.

Boolean encoding is not without a price. Many diagnostic (and SAT) algorithms work on a normalized representation of a model (e.g., CNF). In the Boolean encoding phase, the model loses important locality information (treating the different values of a variable in connection to each other increases the reasoning speed). This problem of "breaking apart" the multi-valued variables is often aggravated later in the normalization as the Boolean encodings of a single multi-value variable can be "spread-through" a model after it has been encoded. Directly using a model represented in multi-valued normal form preserves the locality while a multi-valued diagnostic algorithm retains the simplicity of a Boolean diagnosis algorithm.

The main algorithm, introduced in this paper, is a multi-valued A* search which computes the diagnoses in best-first order, starting with a minimal diagnosis (note that the diagnoses produced subsequently are not necessarily minimal). Thus, not all minimal diagnoses need be computed (the number of all minimal diagnoses can be still exponential in the number of components [Vatan, 2002]).

The method, proposed in this paper, is not the only one which achieves speed-up due to exploiting locality. A complementing approach is to exploit system structure and hierarchy [Feldman *et al.*, 2005; Fattah and Dechter, 1995]. Reasoning in representations which are closer to the "raw" model reduces the number of pre-processing transformation steps and, in general, achieves faster diagnosis times. All these techniques can be combined for achieving diagnosis in real-time for a wide-class of real-world applications.

Surprisingly, there are few publications [Bandelj *et al.*, 2002] concerning the application of non-boolean search algorithms to qualitative reasoning and model-based diagnosis. In the satisfiability field, CAMA [Liu *et al.*, 2003] is a novel extension of the classic DPLL algorithm. The CAMA algorithm uses unit-clause propagation and conflict-learning to increase the performance of the satisfiability checking. Similarly [Frisch and Peugniez, 2001] studies direct non-boolean stochastic local search where a well-known Boolean SAT engine (Walksat) is modified to handle non-boolean problems. The authors of both CAMA and NB-Walksat find their approaches faster for multi-valued SAT instances with increasing domain-size, compared to a Boolean DPLL run on the equivalent Boolean encodings.

Encoding multi-valued problems in the Boolean domain is a technique discussed in many papers. The study of the use of both a Boolean truth maintenance system and the finite domain approach for solving a class of constraint satisfaction problems (CSP) dates back to [de Kleer, 1989]. The latter paper suggests sparse Boolean encoding for finite-domain integer solvers. Multi-valued reasoning is complementary to other locality-based diagnosis search techniques [Provan, 2001]. A study on the use of multi-valued propositional encodings for finite-domain variables and the possibility of combinatorial loss-of-performance due to the increase of the propositional theory size, however, is beyond the scope of that paper.

CSP-based algorithms for model-based diagnosis [Sachenbacher and Williams, 2004; Williams and Ragno, 2004] already consider multi-valued variable domains. An extensive study has been performed on CSP decomposition techniques [Stumptner and Wotawa, 2003] and some empirical results discussed by the same authors in [Stumptner and Wotawa, 2001] show good performance results. The latter decomposition technique results in faster diagnosis time due to tractability achieved by transforming the original problem to an equivalent one with restricted structure. Our technique differs from the above by the fact that it is based on multi-valued propositional search, hence allowing more aggressive optimization by borrowing learning algorithms and heuristics from the satisfiability domain.

The method discussed in this paper can be generalized to almost any approach for propositional reasoning. Multiple-valued decision diagrams (MDD) [Srinivasan *et al.*, 1990] are a natural extension to binary decision diagrams (BDD). To our knowledge the state-of-the-art compilation technique of Darwiche [Darwiche, 2004] has not been generalized to multi-valued propositional logic. Hence an approach similar to ours would be complementary to this method. Similarly, a propositional truth-maintenance system (TMS) (e.g., [Nayak and Williams, 1998]) and Boolean model-based reasoning systems [Frohlich and Nejdl, 1997] can benefit from being able to reason over non-boolean literals.

The type of Boolean encoding influences the performance of the DPLL search and the topic is further studied in [Hoos, 1999]. This paper distinguishes between compact and sparse encodings (also referred to as unary and binary in other papers) and compares the performance of a state-of-the art CSP solver and a Boolean SAT consistency checker. The results show increased performance of the multi-valued approach.

A hybrid finite-domain constraint solver for circuits is suggested in [Parthasarathy *et al.*, 2004]. This approach combines a Boolean DPLL checker and a finite-domain integer CSP solver to allow for more compact problem representation, avoiding the house-keeping constraints imposed by unary or binary encodings (where the number of variables is not a power of 2). This results in a faster solver which has wide application including consistency-based fault diagnosis.

The algorithm described in this paper provides a sound and complete method for computing diagnoses in best-first order. Heuristic functions working on multi-valued representations are provided with additional locality information in comparison to their counterparts working on encoded Boolean models. The extra search dimension which is added by the multi-valued domain of the model variables facilitates faster computation of leading diagnoses. In particular, our method is compared to Boolean algorithms working on both sparse and dense encodings. Sparse encoding leads to combinatorial explosion even with small models, while dense encoding, still slower than the multi-valued approach, imposes difficulties on constructing efficient heuristic functions. All this motivates the use of the direct multi-valued reasoning described below.

The remainder of the paper is organized as follows. In Section 2 we introduce some basic terminology and show a multi-valued consistency checking algorithm, which will be used later in the diagnosis process. In Section 3 we describe the main diagnostic algorithm and illustrate its workings with a small example. Section 4 contains experimental performance results. Finally, conclusion, notes and future works are presented.

2 Multi-Valued Satisfiability

The technique presented in this paper searches for a diagnosis by checking for consistency of a possible health assignment, the system description, and the observation, while discarding the states which are inconsistent. The consistency check is done using a DPLL-based algorithm in the multi-valued domain. The main difference between the multi-valued DPLL and its well-known Boolean counterpart comes from the fact that a multi-valued consistency checking routine can branch on more than two values. Before we show this multi-valued variant of DPLL and discuss the opportunities for its optimization, we introduce the basic terminology necessary for multi-valued satisfiability.

Definition 1 (Multi-Valued Literal)**.** A multi-valued variable $v_i \in V$ takes a value from a finite domain, which is an integer set $D_i = \{1, 2, \ldots, m\}$. A positive multi-valued literal l_j^+ is a Boolean function $l_j^+ \equiv (v_i = d_k)$, where $v_i \in V, d_k \in D_i$. A negative multi-valued literal l_j^- is a Boolean function $l_j^- \equiv (v_i \neq d_k)$, where $v_i \in V, d_k \in D_i$. If not specified, a literal l_j can be either positive or negative.

Note, that a variable v_i can assume at most one value or its complement in D_i. The need for having negative literals comes from the fact that frequently in models, the process of converting to multi-valued CNF results in many negations. For a variable with sufficiently large domains using this notation instead of all complementary values leads to significantly shorter formulae. Next we introduce a multi-valued conjunctive normal form (CNF) which will be our representation for the diagnostic problem[1].

Definition 2 (Multi-Valued CNF)**.** A multi-valued conjunctive normal form is a conjunction of disjunctions of multi-valued literals, that is $C = \sigma_1 \wedge \sigma_2 \wedge \ldots \wedge \sigma_n$ and $\sigma_i = l_{i1} \vee l_{i2} \vee \ldots \vee l_{ik}$ for $i = 1 \ldots n$.

Throughout this text we will also use an alternative notation for a formula in CNF – a clausal set. In this case the clausal set is a set of clauses and each clause is a disjunction of multi-valued literals.

Definition 3 (Multi-Valued Assignment)**.** A multi-valued assignment ϕ or a *multi-valued term* is a conjunction of multi-valued literals, that is $\phi = l_1 \wedge l_2 \wedge \ldots \wedge l_p$.

[1]Translation from multi-valued propositional logic to multi-valued CNF is well-studied and we will omit it for brevity.

Obviously, we can convert a multi-valued assignment to a clausal set in which each clause contains a single literal, and vice-versa, by using the De Morgan's law: $l_1 \wedge l_2 \wedge \ldots \wedge l_p = \neg l_1 \vee \neg l_2 \vee \ldots \vee \neg l_p$. The latter proves useful when, later in Algorithm 2, we have to add an observation represented as an assignment to the clausal set of the system description.

If an assignment contains all the variables in a multi-valued CNF we will call it *full*, otherwise it is *partial*. Next we describe an algorithm, which given a multi-valued CNF C returns $True$ iff there is an assignment (a conjunction of literals) ϕ such that $C \models \phi$.

The worst-case time complexity of Algorithm 1 is exponential of the number of variables in ϕ. Formulae from model-based diagnosis, however, are highly structured and rarely expose this worst-case performance. If we consider all the domain sets $D_i \in D$, the one with the highest cardinality is $d_{\max} = \arg\max_{D_i \in D} |D_i|$, that is $d_{\max}$ is the number of values in the largest variable domain in ϕ. The space complexity of Algorithm 1 is then $O(|V| \times d_{\max})$, where $|V|$ is the number of variables in ϕ.

Algorithm 1 Multi-valued DPLL consistency checking.

```
    function MVSAT($C, V, D, \phi$)
    inputs: $C$, a clausal-set
            $V = \{v_1, v_2, \ldots, v_q\}$, a set of variables
            $D = \{D_1, D_2, \ldots, D_q\}$, a set of domain sets
            $\phi$, an assignment, initially empty

      if ($\phi \leftarrow \phi \wedge$ UNITPROPAGATE($C, \phi$)) $\models \perp$ then
        return $False$
      end if
5:    if $\nexists \sigma \in C, \sigma \wedge \phi \not\models \perp$ then
        return $True$
      end if
      for all $\{v_i \in V, k \in D_i : l \equiv (v_i = k), l \notin \phi\}$ do
        if MVSAT($C, V, D, \phi \wedge l$) $= T$ then
10:         return $True$
        end if
      end for
      return $False$
    end function
```

The workings of Algorithm 1 is very similar to the original Boolean algorithm, except that it branches for every possible value of the selected variable v_i. An important part of the algorithm is the unit propagation, implemented in the UNITPROPAGATE routine. Note that, unlike in the Boolean case, unit-propagation works only for clauses which are unit-open with a free positive literal (in the Boolean case we can assign a value to negative literals as well).

Algorithm 1 is subject to the same optimization techniques as the Boolean DPLL. An important source of speed-up can be conflict learning[2]. In addition to the classical Boolean speed-up methods, an additional source of speedup would be the use of various heuristic techniques which are possible due to the extra search dimension caused by the multiple values. When choosing a value for a given variable, for example, it is possible to select (either dynamically for the remaining values and clauses or statically) the one which will satisfy the biggest number of clauses.

Algorithm 1 has the potential of determining faster satisfiability in comparison to a Boolean DPLL running on an encoding. Consider the example formula $C = (((x = 1) \vee (y = 2)) \wedge ((x = 3) \vee (y \neq 1)))$, with the domains of the variables x and y, $D_x = \{1, 2, 3\}$ and $D_y = \{1, 2\}$, respectively. A sparse Boolean encoding would map $x = 1$ to x_1, $x = 2$ to x_2, $x = 3$ to x_3, $y = 1$ to y_1 and, finally, $y = 2$ to y_2. In addition to that it needs to impose the constraints $x_1 \neq x_2, x_1 \neq x_3, x_2 \neq x_3$ and $y_1 \neq y_2$. The sparsely encoded Boolean formula is $C_s = (x_1 \vee y_2) \wedge (x_3 \vee \neg y_1) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3) \wedge (y_1 \vee y_2) \wedge (\neg y_1 \vee \neg y_2)$. While the multi-valued formula C has 2 variables with 2 and 3 values each and total of 2 clauses, its Boolean encoding C_s has 5 variables and 8 clauses. In the multi-valued case we can determine the satisfiable assignment $(x = 2) \wedge (y = 2)$ in 4 steps only (ignoring the effect of the unit propagation), while in the Boolean encoding we have to perform 6 recursive calls.

3 Diagnosis of Multi-Valued Models

Modeling of physical artifacts and representing them by using first-principles is a topic on its own and in this paper we will assume that a correct model is converted to a multi-valued CNF and what remains to perform is the computationally intensive task of finding all possible explanations of a given observation. In order to suggest such an algorithm we first formalize the notion of a diagnostic problem and diagnosis.

Definition 4 (Diagnostic Problem)**.** A multi-valued diagnostic problem *DP* is defined as the ordered triple $DP = \langle SD, H, OBS \rangle$, where $SD = \langle V, D, C \rangle$ is the set of variables, their domains and a system description represented as a multi-valued CNF, H is a set of health variables such that $H \subset V$ and the observation *OBS* is a variable assignment over some of the variables in $V \setminus H$.

If a device is malfunctioning then assigning nominal values (let us name such a nominal assignment x) to all the health variables of its model *manifests* an inconsistency with the observation, i.e., $OBS \cup SD \cup x \models \perp$. A sound and complete diagnostic algorithm has to find all assignments which *explain* the observation *OBS*, that is $\forall x : x \in X, OBS \cup SD \cup x \not\models \perp$.

Definition 5 (Multi-Valued Diagnosis)**.** A *diagnosis* or *partial diagnosis* for the system $DP = \langle SD, H, OBS \rangle$, $SD = \langle V, D, C \rangle$ is the assignment $x = l_1 \wedge l_2 \wedge \ldots \wedge l_n$ such that $SD \wedge OBS \wedge x \not\models \perp$.

The central problem of model-based diagnosis is that for n health variables we may have as much as 2^n possible diagnoses. In the multi-valued domain, the complexity is even worse due to the multi-valued domains of the variables, i.e., it becomes $O(d^n)$, where d is the cardinality of the biggest domain set. Although the number of diagnoses depends on the model (e.g., if it is a weak or strong fault model) and the

[2]Actually, in the cases of inconsistent input we need a conflict extraction mechanism, the result of which is to be used for pruning the diagnostic tree in Algorithm 2. Due to the limited scope of this paper we will not discuss mechanisms for conflict extraction.

size of the observation, we rarely need *all* diagnoses. An informed search strategy such as A* is a suitable method for computing only these diagnoses which optimize an objective function g. Such an approach allows us to stop the diagnostic algorithm after we have found the first N diagnoses maximizing g.

As a heuristic function, we typically employ a greedy estimator of the probability of an assignment ϕ. Let us assign *a-priori* probabilities to all possible values of all health variables. The a-priori probability function of a health variable h_i we define as $p_i(h_i)$, where $0 \leq p_i(x) \leq 1$, $x \in D_i$ and $\sum_{x \in D_i} p_i(x) = 1$. The probability estimator of a health assignment $\phi = l_1 \wedge l_2 \ldots \wedge \ldots l_k$ is defined as $g(\phi) = p_1(h_1)p_2(h_2)\ldots p_n(h_n)$, where $h_1, h_2, \ldots, h_n$ are *all* health variables $h_i \in H$ and:

$$p(h_i) = \begin{cases} p_i(l_j) & \text{if } l_j^+ \in \phi \\ 1 - p_i(l_j) & \text{if } l_j^- \in \phi \\ \arg\max_{k \in D_i} p_i(k) & \text{otherwise} \end{cases} \quad (1)$$

Next we show the actual multi-valued A* algorithm for model-based diagnosis.

Algorithm 2 Multi-Valued A* Diagnosis.

```
    procedure MVA*(SD, OBS)
    inputs: $SD = \langle V, D, C\rangle$, a system model
            OBS, an observation term
    local variables: $Q$, a priority queue
                     $x$, a health assignment

      PUSH($Q$, INITIALSTATE())
      while ($x \leftarrow$ POP($Q$)) $\not\models \perp$ do
        if MVSAT($SD \cup OBS \cup x, V, D, \emptyset$) then
5:        if ISFULLDIAGNOSIS($h$) then
            OUTPUTDIAGNOSIS($x$)
            ENQUEUESIBLINGS($Q, SD \cup OBS, x$)
          else
            PUSH($Q$,CHILDSTATE($SD \cup OBS, x$))
10:       end if
        else
          ENQUEUESIBLINGS($Q, SD \cup OBS, x$)
        end if
      end while
15: end procedure
```

An implementation of (1) is used for the ordering in the priority queue Q. For the manipulation of the nodes in this queue (each node is an ordinary multi-valued health assignment) we use the standard functions PUSH and POP, the latter returning $\perp$ when the queue is empty.

The queuing of the nodes is performed by the CHILDSTATE and ENQUEUESIBLINGS routines. The former extends a partial assignment with its most probable descendant and the latter pushes on the queue the most probable siblings of each ancestor of the current node. Note that due to the fact that the search is non-systematic we have to keep a list of the visited nodes (this can be organized somewhat more optimally in a trie).

As in many cases diagnostic models can be over- or under-constrained (depending on the modeling technique), Algorithm 2 can be extended with conflict-learning mechanism for pruning parts of the search tree [Williams and Ragno, 2004]. Finding the set of all minimal conflicts is a problem which is itself NP-hard, hence such a technique has no choice but to perform a limited amount of analysis (e.g., through resolution) for finding conflicts of good quality.

We will illustrate the advantages of the multi-valued diagnosis with a simple model of a valve. The health state of the valve we denote as the health variable h, the control variable c denotes the commanded position of the valve, and for the input and output we use i and o, respectively.

The domains of the four variables h, c, i, and o, are $D_h = \{1 \text{ (nominal)}, 2 \text{ (stuck open)}, 3 \text{ (stuck closed)}, 4 \text{ (unknown)}\}$, $D_i = D_o = \{1 \text{ (low pressure)}, 2 \text{ (high pressure)}\}$, and $D_c = \{1 \text{ (open)}, 2 \text{ (close)}\}$, respectively. Additionally, we define the a-priori probability estimator of h as $p(h = 1) = 0.9$, $p(h = 2) = p(h = 3) = 0.045$, and $p(h = 4) = 0.01$. The model is encoded as the multi-valued propositional formula:

$$M = \begin{cases} (h = 1) \wedge (c = 1) \Rightarrow (o = i) \\ (h = 1) \wedge (c = 2) \Rightarrow (o = 1) \\ (h = 2) \Rightarrow (o = i) \\ (h = 3) \Rightarrow (o = 1) \end{cases} \quad (2)$$

Next we convert M to a clausal set and the result is the following:

$$C = \begin{cases} ((c = 2) \vee (i = 2) \vee (o = 1) \vee (h \neq 1)) \\ ((c = 2) \vee (i = 1) \vee (o = 2) \vee (h \neq 1)) \\ ((c = 1) \vee (o = 1) \vee (h \neq 1)) \\ ((i = 2) \vee (o = 1) \vee (h \neq 2)) \\ ((i = 1) \vee (o = 2) \vee (h \neq 2)) \\ ((o = 1) \vee (h \neq 3)) \end{cases} \quad (3)$$

Let us assume an observation $OBS = (c = 1) \wedge (i = 2) \wedge (o = 1)$, that is the valve is stuck-open. The first step of Algorithm 2 would be to check the health assignment $x_1 = (h = 1)$ which has the highest probability estimator $P(x_1) = 0.9$. Algorithm 1 will determine that $SD \cup OBS \cup x_1 \models \perp$, hence we have to try the second-probable health-assignment $x_2 = (h = 2)$ and in this case we have $SD \cup OBS \cup x_2 \not\models \perp$. In this case x_2 is a diagnosis and due to the admissibility of the heuristics involved, we can pronounce x_2 to be the most-likely diagnosis.

Next, we consider the sparse Boolean encoding of C (normally, we would encode M as M_s and convert M_s to CNF which is even worse than encoding directly C to C_s) in (3). To preserve the order in which we generate diagnoses, we assign a probability estimator $p'(h_n)$ to each Boolean health variable h_n encoding a state $h = n$. We define $p(h_n) = p(h = n)$ and $p(\neg h_n) = 1 - p(h = n)$. It is easy to show that such an assignment of the probability estimators preserves the order in which diagnoses are generated.

$$C_s = \begin{cases} (c_2 \vee i_2 \vee o_1 \vee \neg h_1) \wedge (c_2 \vee i_1 \vee o_2 \vee \neg h_1) \\ (c_1 \vee o_1 \vee \neg h_1) \wedge (i_2 \vee o_1 \vee \neg h_2) \\ (i_1 \vee o_2 \vee \neg h_2) \wedge (o_1 \vee \neg h_3) \\ (c_1 \vee c_2) \wedge (\neg c_1 \vee \neg c_2) \wedge (i_1 \vee i_2) \\ (\neg i_1 \vee \neg i_2) \wedge (o_1 \vee o_2) \wedge (\neg o_1 \vee \neg o_2) \\ (h_1 \vee h_2) \wedge (h_1 \vee h_3) \wedge (h_2 \vee h_3) \\ (\neg h_1 \vee \neg h_2) \wedge (\neg h_1 \vee \neg h_3) \wedge (\neg h_2 \vee \neg h_3) \end{cases} \quad (4)$$

A dense encoding of C results in a shorter representation, having the same number of clauses as in the multi-valued

CNF:

$$C_d = \left\{ \begin{array}{l} (h_1 \vee h_2 \vee c \vee i \vee \neg o) \\ (h_1 \vee h_2 \vee c \vee o \vee \neg i) \\ (h_2 \vee h_1 \vee \neg c \vee \neg o) \\ (h_2 \vee i \vee \neg h_1 \vee \neg o) \\ (h_2 \vee o \vee \neg h_1 \vee \neg i) \\ (h_1 \vee \neg h_2 \vee \neg o) \end{array} \right. \qquad (5)$$

While in this example C_d is as compact as C_M, we have twice as many health variables and we would need 8 instead of 4 consistency checks if we were to add an extra health state for h in (2). The dense encodings expose another significant problem with representing the probability estimator as the health variables h_1 and h_2 are not independent. Another disadvantage of the dense encoding is apparent in the cases when the number of states for a multi-valued variable is not a power of 2 (i.e., $k \neq 2^n$ for $n \in \mathbb{N}$, where n is the number of states of a variable v). In this case additional constraints should be added or $2^n - k$ states would use two Boolean encodings per state (such health encodings, however, would result in $2^n - k$ cases in which the same diagnosis is computed twice, necessitating the storing of the already generated diagnoses).

A Boolean A* diagnosis[3] on the sparse-encoded model has to perform 2^3 consistency checks in (4) for finding all diagnoses, while Algorithm 2 computers all of them with 3 checks only (in addition to that multi-valued consistency checking is more efficient as we have seen in Section 2). With dense encodings we need 4 consistency checks which is still less efficient than the multi-valued case.

The performance difference is better illustrated when we are interested in the first (most-likely) diagnosis only. In this case the multi-valued algorithm needs 2 consistency checks over a clausal set comprising 6 clauses, the sparse encodings allow computation of a leading diagnosis with 5 checks over 18 clauses and the dense encodings do not facilitate an implementation of a heuristic function preserving the probability order. The number of clauses in C_s is 18 due to the extra inequality constraints which additionally delays the Boolean reasoning. As we are going to show in the next section, these differences translate to significant savings (in favor of the multi-valued approach) with bigger models.

4 Experimental Performance Evaluation

To demonstrate the improved performance of the multi-valued solver we have compared it experimentally with the performance of a Boolean solver for sparse and dense encoded models. In these experiments we use the diagnosis time t as performance metric. This is the processing time required to generate N diagnoses, for given *OBS*. It is measured by the diagnosis engines as CPU time in milliseconds. All the experiments described in this paper are performed on a 3 GHz Pentium IV CPU.

The algorithm discussed in this section were performed as a part of the LYDIA model-based diagnosis toolkit[4]. The benchmark models and test-vectors are available upon request. The multi-valued polycell models, discussed below were synthetically generated. Our experimentation could further benefit from the existence of a scalable multi-valued benchmark for model-based diagnosis.

For a multi-valued problem and its sparse and and dense encodings, we denote t with t_m, t_s, and t_d respectively. The speedups s_s and s_d of the multi-valued search over the Boolean sparse and dense encodings respectively, are calculated as $s_s = t_s/t_m$ and $s_d = t_d/t_m$. Let W denote the non-observable, non-health variables, $W = V \setminus (\mathit{OBS} \cup H)$. We investigate t, s_s and s_d in relation to the domain size $|D_i|$ and model complexity.

As discussed in the valve example, $|D_i|$ and the type of encoding determine the number of clauses and consistency checks which affect t. Hence we investigate the relation between t and $|D_i|$. We use the well-known synthetic, integer domain Polycell model [de Kleer and Williams, 1987] which allows for practical and meaningful variation of $|D_i|$. Let $|D_i| = d_H$, $\forall h_i \in H$ and $|D_i| = d_W$, $\forall w_i \in W$. With the Polycell we perform experiments for $d_H = 2, 3, \ldots, 9$ and $d_W = 2, 3, \ldots, 42$, for nominal *OBS* with N equal to the maximum number of consistent solutions K.

We perform similar performance experiments on real-world models to investigate whether the expected improved performance of the multi-valued approach also holds for practical applications. We use nine models of variations of ASML wafer scanner subsystems and one of NASA's X-34 propulsion system. In contrast to Polycell, these models have varying values of $|D_i|$ for H and W. Therefore we cannot investigate a one-to-one relationship of t with $|D_i|$. Instead, we investigate the dependencies between t and the model complexity. We use S, the number of all possible value assignments of H and W, as measure for this complexity. Let $S_H = \prod_{h_i \in H} |D_i|$, $h_i \in H$, $S_W = \prod_{w_i \in W} |D_i|$, $w_i \in W$, and $S = S_H S_W$. Table 1 shows these numbers for all real-world models and variations.

Model	S_H	S_W	S
ASML1A1	$2^2 4^1$	$2^2 3^7$	$1.39E+05$
ASML1A2	$2^2 4^1$	$2^4 3^1 3$	$4.08E+08$
ASML1A3	$2^2 4^1$	$2^6 3^1 9$	$1.19E+12$
ASML1B1	$2^6 4^3$	$2^2 3^7$	$3.58E+07$
ASML1B2	$2^{12} 4^6$	$2^4 3^{13}$	$4.28E+14$
ASML1B3	$2^{18} 4^9$	$2^6 3^{19}$	$5.11E+21$
ASML2A	2^{10}	$2^{15} 3^1 4^{10}$	$1.06E+14$
ASML2B	2^{10}	$2^{30} 3^2 4^{20}$	$1.09E+25$
ASML2C	2^{10}	$2^{45} 3^3 4^{30}$	$1.12E+36$
X-34	$2^{26} 4^{12}$	3^{20}	$3.93E+24$

Table 1: S_H, S_W, and S for 10 real-world models and variations.

We consider three scenarios with *OBS* caused by nominal, single fault, and double faults for $N = 1$ and $N = \min(|H|, K)$, respectively. Because we do not currently have a probability heuristic in place for the dense encoding, its diagnoses is unsorted. Hence comparison with the sorted multi-valued diagnosis for $N < K$, is not always correct.

[3]Note that (4) can be considered as a multi-valued diagnosis problem with two values per variable.

[4]The LYDIA package for model-based fault diagnosis can be downloaded from `http://www.fdir.org/lydia/`.

For the Polycell model Figure 1 shows a logarithmic plot of t against d_W with $d_H = 2$, for all encodings. For $d_W \to 42$, $s_s \approx 6$ and $s_d \approx 1$, the latter indicating similar performance. The down-ward spikes in the dense encoding plots are due to the efficient dense encoding for $\log_2 d_W \in \mathbb{N}$. Experiments performed with non-nominal or fewer observations show similar results.

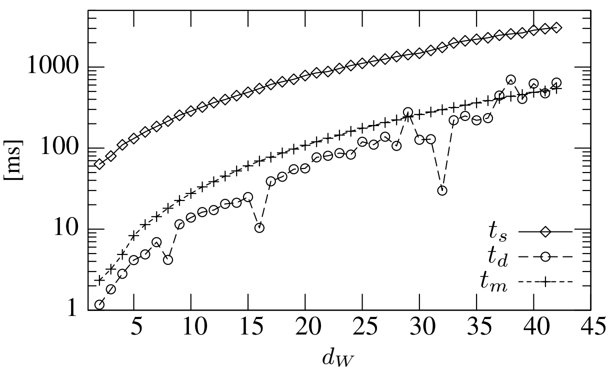

Figure 1: Time for computing of all diagnoses for Polycell models with $d_H = 2$ and a nominal observation vs. variable d_W.

Similarly Figure 2 shows the results for $d_H = 3$. For $d_W \to 42$, $s_s \approx 26$ and $s_d \approx 3$. Thus, except for the spikes at $d_W = 16$ or $d_W = 32$ the multi-valued approach now also clearly outperforms the dense encoding. This is due to the inefficiency of dense encodings for $\log_2 d_H \notin \mathbb{N}$.

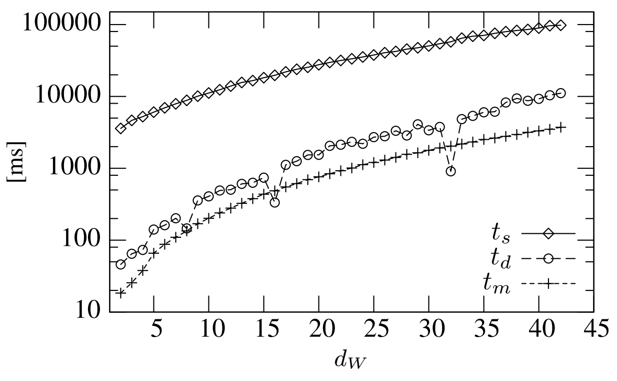

Figure 2: Time for computing of all diagnoses for Polycell models with $d_H = 3$ and a nominal observation vs. variable d_W.

Figure 3 shows a double logarithmic plot of t against d_H. We omit sparse encodings because t_s for $d_H > 3$ becomes prohibitively large for practical experiments. The increase of t_d is exponential $\propto \lceil \log_2 d_H \rceil$, hence the staircase shape of this plot. Let $\gamma_d(d_H)$ be the increase according to,

$$\gamma_d(d_H) = \frac{t_d(d_H | \lceil \log_2 d_H \rceil = k)}{t_d(d_H | \lceil \log_2 d_H \rceil = k - 1)}, k = 2, 3, 4 \qquad (6)$$

For $k = 2, 3, 4$, $\gamma_d(d_H) \approx 17, 21, 31$. The straight line for multi-valued encodings agrees to $t \propto {d_H}^5$. The plot shows that for $d_H < 9$ and $\log_2 d_H \uparrow \lceil \log_2 d_H \rceil$, $s_d < 1$, i.e., better performance for dense encoding.

As, t_m and t_d for $d_H > 9$ become prohibitively large for experiments we need to extrapolate their relations with d_H to compute s,

$$s_d = \frac{\gamma_d(d_H)^{\lceil \log_2 d_H \rceil}}{{d_H}^5} \qquad (7)$$

If, e.g., we consider a conservative approach and assume $\gamma_d(d_H) = 31$, then $0.8 \leq s_d \leq 23.9$, for $d_H \leq 256$. If, as expected from the experimental results $\gamma_d(d_H)$ continues to increase, then for $\gamma_d(d_H) > 2^5$, $1.0 < s_d$ and the speedup of multi-valued over dense encoding will not have an upper bound. Thus multi-valued encoding also has unbounded speedup for dense encodings.

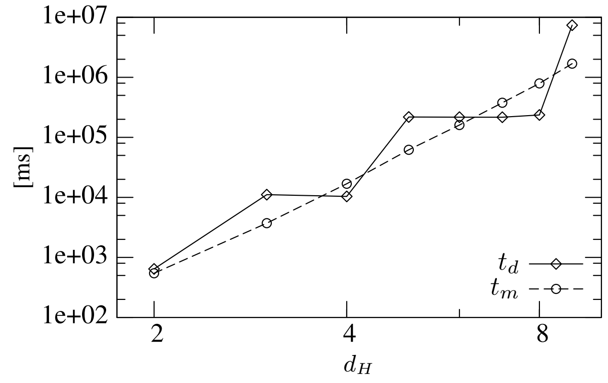

Figure 3: Double logarithmic plot of time for computing of all diagnoses for multi-valued and dense encoded Polycell models with $d_W = 42$ and a nominal observation vs. variable d_H.

For the real-world models, Figure 4 shows the double logarithmic plot for t vs. S for $N = 1$ for the multi-valued solver for nominal, single, and double faults. As expected, it shows an increasing trend for t vs. S, and for increased cardinality of the faults. Figure 5 shows s_s and s_d. Note that data points above the thick line $s = 1$ indicate an actual speedup of the multi-valued approach. In the nominal case, $s_s < 1$ which is caused by a larger overhead of the multi-valued approach for $N = 1$. The effect of this overhead disappears for single and double faults where $10^1 < s_s < 10^4$. Because of the omitted probability heuristic mentioned earlier, the analysis of s_d is less straightforward. We note that in roughly half the cases the speedup is similar to the sparse encoding.

Figures 6 and 7 show t and s for $N = \min(|H|, K)$. As $N \geq 1$ the initial overhead effect for the multi-valued approach is amortized over the multiple diagnoses resulting in $10^1 < s_s < 10^4$ for nominal, single, and double faults. For s_d, it is interesting to see that the effect of the omitted probability heuristic is most apparent for the double faults. This can be explained from the fact that in these cases the solver generates many-fault solutions with low probability faster than the multi-valued solver generates true double-fault solutions. The latter have higher t because of more constrained consistency checking.

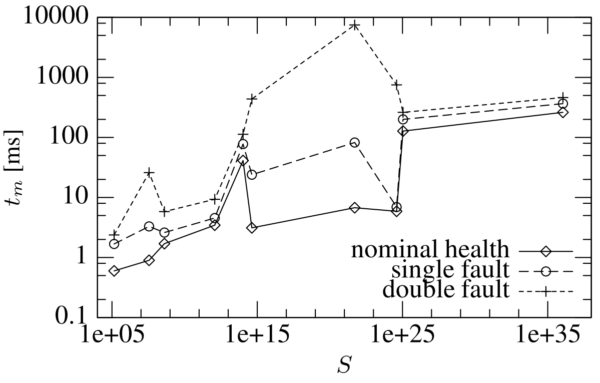

Figure 4: Double logarithmic plot of diagnosis time with $N = 1$ and observations consistent with different fault cardinalities vs. variable model complexity.

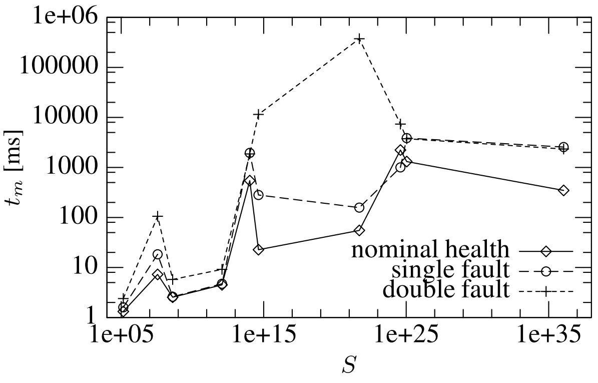

Figure 6: Double logarithmic plot of diagnosis time with $N = \min(|H|, K)$ and observations consistent with different fault cardinality vs. variable model complexity.

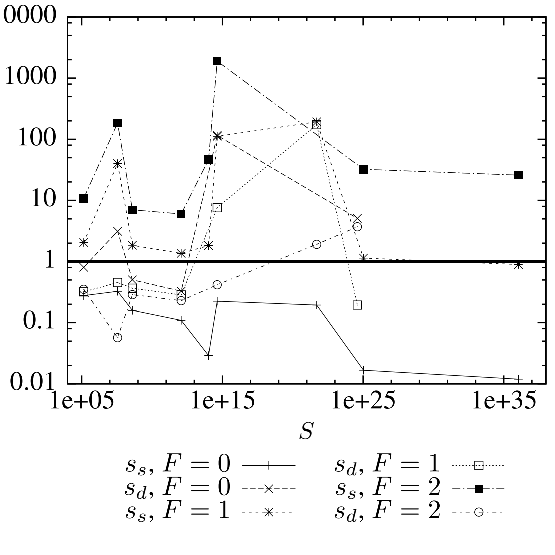

Figure 5: Double logarithmic plot of speedup with $N = 1$ and observations consistent with nominal health ($F = 0$), single fault ($F = 1$), and double fault ($F = 2$) vs. variable model complexity.

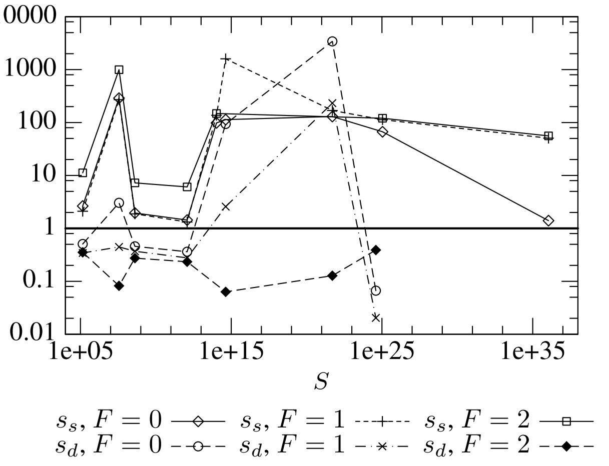

Figure 7: Double logarithmic plot of speedup with $N = \min(|H|, K)$ and observations consistent with nominal health ($F = 0$), single fault ($F = 1$), and double fault ($F = 2$) vs. variable model complexity.

In summary of the experiments, the speedup of multi-valued approach over the sparse encoding is demonstrated clearly both in relation to the domain size and number of states. Speedups of 10^2 are readily achieved. The same conclusion holds for dense encodings as far as the domain size experiments are concerned. Especially for larger domain sizes the speedup is considerable. For smaller domain sizes closer proximity to $2^i, i \in \mathbb{N}$ means better performance for the dense encodings. Because of the lacking probability heuristic, the relation of s_d with S remains somewhat unclear, despite the fact that $s_d > 5$ in many cases.

5 Conclusion

This paper introduces a new algorithm for computing diagnoses which works directly on the multi-valued representation of a model. The sound and complete algorithm comprises a DPLL-based multi-valued consistency checking and a multi-valued A* search. The two routines eliminate the need of model encodings, thus preventing loss of locality information which often leads to performance degradation.

In contrast to dense Boolean encoding, the multi-valued A* algorithm allows an intuitive assignment and interpretation of a-priori probabilities, which combined with the greedy A* search, described in this paper, allows for a precise control over the termination criteria for the diagnostic computation. This allows the generation of only these leading diagnoses that contain significant probability mass (thus turning the search into incomplete). While sparse Boolean encoding is more suitable than dense encoding for heuristic search based on a-priori probability, the combinatorial explosions caused by the introduction of new variables makes it suitable for the tiniest diagnosis problems only.

We have empirically compared the performance of the algorithm to sparse and dense Boolean encodings. These experimental results confirm our analysis and show that the multi-valued search outperforms both types of encoding, in particular being two orders of magnitude faster than sparse Boolean encoding. While dense encoding is faster than sparse encoding (but still slower in comparison to the multi-valued approach), it is less amenable to heuristics based on a-priori health probability estimation, widely used in model-based diagnosis.

In future work, we aim to address this estimator problem, allowing the three methods to be compared not only in the cases of generating all diagnoses but also when computing the first N leading diagnoses. In this case we would be able to analyze how the probability assignment influences the diagnostic search. Last, we note the need of representative multi-valued benchmarks which would enhance the experimental results of this paper.

Furthermore, we envision our algorithm used in combination with other reasoning-methods which result in improved performance by using locality. Analysis [Ramesh *et al.*, 1997] and experience has shown that the higher-level reasoning engine we use (by higher-level we mean a diagnostic engine which uses a model representation closer to the original model), the faster performance results we get. In the future, we would be interested in combining, e.g., non-clausal search methods, hierarchical search [Feldman *et al.*, 2005], and the multi-valued approach described here for a more efficient model-based diagnosis and related diagnostic reasoning.

Acknowledgments

We extend our gratitude to the anonymous reviewers for their valuable feedback.

References

[Bandelj *et al.*, 2002] A. Bandelj, I. Bratko, and D. Šuc. Qualitative simulation with CLP. In *Proc. QR'02*, 2002.

[Darwiche, 2004] Adnan Darwiche. New advances in compiling CNF into DNNF. In *Proc. ECAI'04*, pages 328–332, 2004.

[de Kleer and Williams, 1987] J. de Kleer and B. Williams. Diagnosing multiple faults. *JAI*, 32(1):97–130, 1987.

[de Kleer, 1989] J. de Kleer. A comparison of ATMS and CSP techniques. In *Proc. IJCAI'89*, pages 290–296, 1989.

[Fattah and Dechter, 1995] Yousri El Fattah and Rina Dechter. Diagnosing tree-decomposable circuits. In *IJCAI'95*, pages 1742–1749, 1995.

[Feldman *et al.*, 2005] A. Feldman, A.J.C van Gemund, and A. Bos. A hybrid approach to hierarchical fault diagnosis. In *Proc. DX'05*, pages 101–106, 2005.

[Frisch and Peugniez, 2001] Alan Frisch and T. Peugniez. Solving non-boolean satisfiability problems with stochastic local search. In *Proc. IJCAI'01*, pages 282–290, 2001.

[Frohlich and Nejdl, 1997] Peter Frohlich and Wölfgang Nejdl. A static model-based engine for model-based reasoning. In *Proc. IJCAI'97*, pages 466–473, 1997.

[Hoos, 1999] H. Hoos. SAT-encodings, search space structure, and local search performance. In *Proc. IJCAI'99*, pages 296–303, 1999.

[Liu *et al.*, 2003] C. Liu, A. Kuehlmann, and M. Moskewicz. CAMA: A multi-valued satisfiability solver. In *Proc. ICCAD'03*, pages 326–333, 2003.

[Nayak and Williams, 1998] Panduarng Nayak and Brian Williams. Fast context switching in real-time propositional reasoning. In *Proc. AAAI'98*, pages 50–56, 1998.

[Parthasarathy *et al.*, 2004] G. Parthasarathy, M. Iyer, K.T. Cheng, and L. Wang. An efficient finite-domain constraint solver for circuits. In *Proc. DAC'04*, pages 212–217, 2004.

[Provan, 2001] G. Provan. Hierarchical model-based diagnosis. In *Proc. DX'01*, pages 167–174, 2001.

[Ramesh *et al.*, 1997] A. Ramesh, G. Becker, and N. Murray. CNF and DNF considered harmful for computing prime implicants/implicates. *JAR*, 18(3):337–356, 1997.

[Sachenbacher and Williams, 2004] Martin Sachenbacher and Brian Williams. Diagnosis as semiring-based constraint optimization. In *Proc. ECAI'04*, pages 873–877, 2004.

[Sgarlata and Winters, 1997] P. Sgarlata and B. Winters. X-34 propulsion system design. In *Proc. AIAA Joint Propulsion Conference and Exhibit*, 1997.

[Srinivasan *et al.*, 1990] Arvind Srinivasan, Timothy Kam, Sharad Malik, and Robert Brayton. Algorithms for discrete function manipulation. In *Proc. ICCAD'90*, pages 92–95, 1990.

[Stumptner and Wotawa, 2001] Markus Stumptner and Franz Wotawa. Diagnosing tree-structured systems. *JAI*, 127(1):1–29, 2001.

[Stumptner and Wotawa, 2003] Markus Stumptner and Franz Wotawa. Coupling CSP decomposition methods and diagnosis algorithms for tree structured systems. In *Proc. IJCAI'03*, pages 388–393, 2003.

[Vatan, 2002] F. Vatan. The complexity of the diagnosis problem. Technical Report NPO-30315, Jet Propulsion Laboratory, California Institute of Technology, 2002.

[Williams and Ragno, 2004] Brian Williams and R. Ragno. Conflict-directed A* and its role in model-based embedded systems. *JDAM*, 2004.