A Hybrid Approach to Hierarchical Fault Diagnosis

Alexander Feldman[1,2] , **Arjan van Gemund**[2] and **André Bos**[1]
[1]Science & Technology BV, P.O. Box 608, 2600 AP, Delft, The Netherlands
Tel.: +31 15 2629889, Fax: +31 15 2629567, e-mail: {feldman,bos}@science-and-technology.nl
[2]Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Mekelweg 4, 2628 CD, Delft, The Netherlands
Tel.: +31 15 2786144, Fax: +31 15 2786632, e-mail: a.j.c.vangemund@ewi.tudelft.nl

Abstract

For many systems of realistic size the computational complexity of complete model-based diagnosis is prohibitive. In order to overcome this computational hurdle, approximation-based approaches do not solve the complete problem, but compute only the most likely candidates. In this paper we take an alternative approach and investigate the speedup of the diagnosis process by exploiting the hierarchy as is present in well-engineered systems. The approach comprises a compile-time and a run-time step. In the first step, a hierarchical CNF representation of the system is compiled to hierarchical DNF of adjustable depth. In the second step, the diagnoses are computed from the (partially) hierarchical DNF and the actual observations. As in the second step we may apply a complete or an approximation algorithm, our approach can be viewed as complementary towards earlier work on speeding up the diagnosis process. Our hierarchical approach allows large models to be diagnosed, where compile-time investment directly translates to run-time speedup. The benefits of our approach are illustrated by using a 2-bit and 16-bit Full Adder, and the LOFAR Array Telescope which is currently being constructed in The Netherlands. Even for these modestly hierarchical models the speedup compared to traditional approaches ranges in the hundreds.

1 Introduction

Fault diagnosis is a computationally very demanding problem. In this paper we study mechanisms for exploiting hierarchy in a divide-and-conquer approach to significantly lower the computational cost. The potential of the hierarchical approach is to reduce the complexity of the diagnosis computation to that of the biggest subsystem in a model (e.g., in a system compromising non-connected subsystems the time for computing diagnosis can be reduced to the sum of the times for diagnosing each subsystem separately). For a class of decomposable systems this may lead to substantial savings in the diagnosis complexity. In this paper we present a hierarchical approach where we demonstrate that real-world systems may have sufficient hierarchy and subsystem independence for such significant gains to be made.

Exploiting hierarchy has been the subject of much work. In [Stumptner and Wotawa, 2003], a diagnosis problem is represented as a Constraint Satisfaction Problem (CSP). In this framework the issues of model decomposition and hierarchical diagnosis are discussed. The problem of discovering hierarchies is also treated in [Provan, 2001], while [Mozetič, 1991] discusses methods for hierarchical abstraction. Implicit system structure to speedup diagnosis is used in [Darwiche, 1998]. Our technique differs in the fact that it uses explicitly specified system decomposition and in the way it combines combinatorial and probabilistic approaches.

Our hybrid approach comprises a compile-time and a run-time step, both of which exploit the hierarchy of the models. The input model is assumed to be represented in terms of a hierarchical CNF (a conjunctive normal form is a conjunction of clauses, where a clause is a disjunction of literals and a literal is a negated or unnegated variable). In the first step the model is compiled to DNF (a disjunctive normal form is a disjunction of terms, where a term is a conjunction of literals) where significant speedup is obtained, compared to traditional CNF to DNF compilation (e.g., we achieve a speedup of 549 even for a 2-bit adder). This conversion to DNF may still take exponential time, but we have to do this only once as only the fixed part of the model is involved.

A special feature of the hierarchical CNF compilation is that it can also compile to *hierarchical* DNF where the depth of the hierarchy can be varied between fully hierarchical and fully expanded ("flat", i.e., ordinary DNF). This feature allows to avoid the combinatorial explosion that occurs for very large models that are not amenable to full compilation. In this "short-cut" mode an adjustable part of the computational work is transferred from compile-time to run-time.

In the second step the compiled model, together with the observations is solved, generating the diagnoses. The diagnosis approach taken in this paper is based on an informed search algorithm that exploits the hierarchy as far as still present in the pre-processed model. In particular, we have chosen A* using an admissible heuristics based on a-priori health state probabilities such as in Conflict-Directed A* (CDA* [Williams and Ragno, 2004]), but other informed strategies are possible. Although the choice for CDA* would further improve performance, in our aim to assess the poten-

tial of hierarchy we have focused on adapting the more simple A* algorithm to our hierarchical framework.

We have compared the performance of the hierarchical algorithm with the performance of a traditional A* diagnosis search. Depending on the structure of the problem we observe speedups up to 9×10^5. In the worst case the hierarchical approach performs similar to traditional A*, while speedup of $10^2 - 10^5$ is measured for well-decomposed problems.

The rest of the paper is organized as follows. In Section 2 we introduce concepts and terminology in traditional model-based diagnosis. In Section 3 we present our compile-time and run-time diagnosis algorithms. In Section 4 we present the performance results. Finally, conclusions and notes for future work are presented.

2 Non-Hierarchical Diagnosis

We will base the hierarchical diagnosis approach on the well-known model-based diagnosis formalisms introduced by [de Kleer and Williams, 1987]. These classical approaches in general use one level of hierarchy, i.e., the set of components of which the overall system is built up.

Definition 1 (System). A diagnostic problem *DP* is defined as the ordered triple $DP = \langle SD, COMPS, OBS \rangle$, where *SD* is a set of propositional sentences describing the behavior of the system, *COMPS* is a set of components, contained in the system, and *OBS* is a term stating an observation over some set of "measurable" variables in *SD*.

For brevity, we refer to the classical diagnosis approach as "flat", i.e., non-hierarchical. In the latter technique we have for each component $c \in COMPS$ a propositional variable h_c that represents a normally functioning component, c. We will call the variables h_c *health variables* and every instantiation of $\bigwedge_{c \in COMPS} h_c$ a *health state*.

In this paper we use consistency-based diagnosis as opposed to abductive diagnosis. Next, we proceed with a more precise definition of diagnosis.

Definition 2 (Diagnosis). A *diagnosis* or *partial diagnosis* for the system $DP = \langle SD, COMPS, OBS \rangle$ is a set $D \subseteq COMPS$ such that:

$$SD \wedge OBS \wedge \left[\bigwedge_{c \in D} \neg h_c \right] \wedge \left[\bigwedge_{c \in (COMPS \setminus D)} h_c \right]$$

is consistent.

A diagnosis D is a *minimal* or *kernel diagnosis* if no other diagnosis D', such that $D' \subset D$, exists. An informed search, such as the A* algorithm used by us, 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]).

As we have already mentioned in Section 1, the diagnosis can be split in two phases: a compilation step and a run-time step. The compilation step constitutes a conversion of the system description into DNF. From DNF diagnosis is straightforward as described by Proposition 1.

Proposition 1. Let $DP = \langle SD, COMPS, OBS \rangle$, be a system. Then D is a *partial diagnosis* of *DP* iff the conjunction of all the health variables of the elements in D is an *implicant* of $SD \wedge OBS$.

From Proposition 1 it follows that to perform diagnosis, it is enough to convert *SD* to DNF and to check each term in the resulting DNF for consistency with *OBS*. Furthermore, in order to receive all the minimal diagnoses, it is necessary to compute a minimal cover of the prime implicants of $SD \wedge OBS$.

In this paper we assume that all the components are described in CNF[1]. The non-hierarchical approach is illustrated by diagnosing a small circuit, shown in Figure 1.

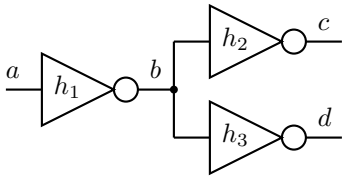

Figure 1: A circuit consisting of three inverters.

For a weak-fault model the corresponding propositional system is given by:

$$SD = \begin{cases} (\neg h_1 \vee a \vee b) \wedge (\neg h_1 \vee \neg a \vee \neg b) \\ (\neg h_2 \vee b \vee c) \wedge (\neg h_2 \vee \neg b \vee \neg c) \\ (\neg h_3 \vee b \vee d) \wedge (\neg h_3 \vee \neg b \vee \neg d) \end{cases} \quad (1)$$

In (1), the health status of the component set $COMPS = \{I_1, I_2, I_3\}$ is given by the health variables h_1, h_2 and h_3. Converting *SD* to DNF results in: $\phi = (\neg h_1 \wedge \neg h_2 \wedge \neg h_3) \vee (\neg a \wedge b \wedge \neg h_2 \wedge \neg h_3) \vee (a \wedge \neg b \wedge \neg h_2 \wedge \neg h_3) \vee (\neg b \wedge c \wedge \neg h_1 \wedge \neg h_3) \vee (b \wedge \neg c \wedge \neg h_1 \wedge \neg h_3) \vee (\neg b \wedge d \wedge \neg h_1 \wedge \neg h_2) \vee (b \wedge \neg d \wedge \neg h_1 \wedge \neg h_2) \vee (\neg b \wedge c \wedge d \wedge \neg h_1) \vee (b \wedge \neg c \wedge \neg d \wedge \neg h_1) \vee (\neg a \wedge b \wedge \neg d \wedge \neg h_2) \vee (a \wedge \neg b \wedge d \wedge \neg h_2) \vee (\neg a \wedge b \wedge \neg c \wedge \neg h_3) \vee (a \wedge \neg b \wedge c \wedge \neg h_3) \vee (\neg a \wedge b \wedge \neg c \wedge \neg d) \vee (a \wedge \neg b \wedge c \wedge d)$.

Consider $OBS = a \wedge \neg c \wedge d$ over the observable variables a, c and d. Instantiating ϕ with *OBS* gives us: $\phi \wedge OBS \models (\neg h_1 \wedge \neg h_2 \wedge \neg h_3) \vee (\neg b \wedge \neg h_2 \wedge \neg h_3) \vee (b \wedge \neg h_1 \wedge \neg h_3) \vee (\neg b \wedge \neg h_1 \wedge \neg h_2) \vee (\neg b \wedge \neg h_2)$. Two implicants of $\phi \wedge OBS$ form minimal diagnoses; these are $D_1 = \{\neg h_2\}$ and $D_2 = \{\neg h_1, \neg h_3\}$ (and are the first ones to be generated by A* for equal a priori health probabilities). Note, that in this particular case $D_3 = \{\neg h_1, \neg h_2, \neg h_3\}$ is also a diagnosis but it is not minimal.

The conversion to DNF in the above example can be accomplished off-line, thus demonstrating a non-strict compilation approach. It is beyond the scope of this paper to discuss compilation techniques, [Liberatore, 1998], but we note that the above compilation step is similar to the first step of our hierarchical approach.

3 Hierarchical Diagnosis

In this section we present our hybrid, hierarchical approach. First we introduce an algorithm for the compilation of hierarchical models. Next we present an algorithm for hierarchical A* diagnosis of the compiled model.

[1]For propositional Wff to CNF cf. [Tseitin, 1983].

In order to present the algorithms we first introduce the notion of a hierarchical system in terms of a tree-like data structure which includes ordinary "flat" systems in its nodes.

Definition 3 (Hierarchical System). A hierarchical system is a rooted, edge-labeled, acyclic multidigraph $H = \langle V, \rho, E \rangle$, where every node V_i, $V_i \in V$, contains a knowledge base SD_i and a set of components $COMPS_i$. The multidigraph is such that $COMPS_1 \cap COMPS_2 \cap \ldots \cap COMPS_n = \emptyset$. The root node is marked by ρ and the labels of the edges in E are maps $f : SD_i \rightarrow SD_j$ between the literals in the knowledge bases represented by the nodes V_i and V_j.

Furthermore, we define hierarchical CNF and hierarchical DNF as hierarchical representation systems with the propositional knowledge base of each node $V_i \in V$ in CNF or DNF respectively. A *hierarchical diagnosis problem* is an ordered pair $HP = \langle H, OBS \rangle$ where OBS is a term over some observable variables in H. The size of a hierarchy can be defined in terms of the size of the knowledge-bases in the nodes of the hierarchy: $|H| = \sum_{e \in E} |SD_e|$, where $|SD_e|$ is the size (e.g., if SD is DNF, the number of terms) of the knowledge-base in the node in which edge e terminates.

In our approach we also consider the depth of a hierarchy d, which is one of its fundamental parameters.

Definition 4. The depth d of a hierarchy $H = \langle V, \rho, E \rangle$ is the number of nodes in the longest path from ρ to any node $v \in V$ such that v is of outdegree 0.

3.1 Hierarchical CNF to DNF Transformation

In this section we present the first part of our hybrid technique. Algorithm 1 modifies its input (a hierarchical CNF), producing a hierarchical DNF of adjustable size. As we will experimentally confirm in Section 4, the more processing time we spend at this phase, the more speedup we obtain at the diagnosis in the second, run-time, part of our technique.
The nested subroutine FLATTENNODE converts a CNF hierarchy to a flat DNF by converting each node to DNF and then multiplying the nodes alongside the multidigraph edges. The function COMPILEDNF transform a node's CNF to DNF. This function can be implemented using a slightly modified satisfiability checker [Davis and Putnam, 1960], modified to enumerate all possible instantiations, or alternatively adding iteratively each negated assignment to the original formula until no more consistent instantiations exist.

We use Algorithm 1 in three configurations, depending on the parameter t. Let d be the initial depth of the hierarchy H. If $t = d$, then FLATTENNODE will never descend recursively, and after FLATTEN finishes, H will be the original hierarchy with each of its node converted from CNF to DNF. The result is hierarchical DNF which we denote DNF/H. On the other extreme we may invoke FLATTEN with parameter $t = 1$. In this case FLATTEN will *fully flatten* H, i.e., the result will have one node only. The result of FLATTEN in this configuration we denote as DNF/F. In most of the cases we have $1 < t < d$. The result of FLATTEN, then, will be a *partially flattened* DNF (DNF/P) as the depth of the DNF/P is t.

To demonstrate Algorithm 1 we consider a small CNF hierarchy H consisting of three nodes (V_1, V_2 and V_3) and two edges ($e_1 = \langle V_1, V_2 \rangle$ and $e_2 = \langle V_1, V_3 \rangle$). Let V_1, V_2 and V_3 contain respectively $SD_1 = (\neg h_1 \vee z) \wedge (x \vee z)$, $SD_2 = \neg h_2 \wedge \neg x$ and $SD_3 = (\neg h_3 \vee z) \wedge y$. In this example we will produce DNF/F, hence we invoke FLATTEN with $t = 1$. In this case COMPILEDNF will be immediately invoked on ρ and FLATTEN will never recursively descend. The workings of FLATTENNODE on ρ are described next. The formula SD_1 in node V_1 needs to be converted to DNF, and at the first invocation of FLATTENNODE we get $P = \{\neg h_1 \wedge x, z\}$. The function is invoked recursively for e_1 and as in the second node SD_2 is already in DNF the result is: $Q = \{\neg h_2 \wedge \neg x\}$. Multiplying P and Q results in a set with one consistent term: $R = \{\neg h_2 \wedge \neg x \wedge z\}$. At the second recursive call of FLATTENNODE we convert $SD_3 = (\neg h_3 \vee z) \wedge y$ to DNF which results in $Q = \{\neg h_3 \wedge y, y \wedge z\}$. Multiplying the partial result R by P for the second time results in the final set of terms $R = \{\neg h_2 \wedge \neg h_3 \wedge \neg x \wedge y \wedge z, \neg h_2 \wedge \neg x \wedge y \wedge z\}$.

The first and the second multiplications in the above example are performed in two steps each. The whole transformation takes four steps, which is obviously an improvement over a brute-force enumeration of all possible instantiations over the variables h_1, h_2, h_3, x, y, and z in the flat CNF formula produced from the hierarchy H. In Section 4.1 we show the speedup potential of the algorithm compared to non-hierarchical compilation.

Algorithm 1 Hierarchical model compilation.

```
procedure FLATTEN(H, N, t, r)
    inputs: H = ⟨V, ρ, E⟩, hierarchy
            N, the current node, N ∈ V, initially ρ
            t, integer, maximal depth
            r, integer, current depth, initially 1
    local variables: SD, the CNF in N
    function FLATTENNODE(H, K) returns a DNF
        inputs: H = ⟨V, ρ, E⟩, hierarchy
                K, the current node, K ∈ V
        local variables: SD_1, the CNF in K
                         P, Q, term sets
                         R, term set, initially ∅
        P ← COMPILEDNF(SD_1)
        for all {e ∈ E : e = V → L} do
5:          Q ← FLATTENNODE(H, L)
            for all {p ∈ P, q ∈ Q : p ∧ q ⊭ ⊥} do
                R ← R ∪ {p ∧ q}
            end for
        end for
10:     return R
    end function
    if t = r then
        SD ← FLATTENNODE(H, N)
        E ← E \ {e ∈ E : e = V → M}
15: else
        SD ← COMPILEDNF(SD_1)
    end if
    for all {e ∈ E : e = V → M} do
        FLATTEN(H, M, t, r + 1)
20: end for
end procedure
```

3.2 A* Search in a Hierarchical DNF

In this section we present the hierarchical diagnosis algorithm, which is based on A*. We assume that components failures are independent and use the *a priori* probability of a fault term to guide a heuristic search for the most likely diagnosis. The heuristic function for getting to a node containing a term σ in a tree constructed from a hierarchy with an equivalent DNF, ϕ, is $f(\sigma) = P_1(h_1)P_2(h_2)\dots P_n(h_n)$, where $h_1, h_2, \dots, h_n$ are all the health variables in ϕ. In this case the probability of h_i being true if h_i is in σ is $P(h_i)$ and $P_i(h_i) = 1 - P(h_i)$ if $\neg h_i$ is in σ. If neither h_i nor $\neg h_i$ are present in σ, $P_i(h_i) = \max(P(h_i), 1 - P(h_i))$. Traversing the nodes of the hierarchy while using $f(\sigma)$ as a heuristics, allows us to construct the A* algorithm shown below.

Algorithm 2 A* search in a hierarchical DNF dictionary.

```
    procedure HIERARCHICALDIAGNOSE(H)
        inputs: H, hierarchical node
        local variables: Q, priority queue
                         s, c, terms

        PUSH(Q, INITIALSTATE(H))
        while SIZE(Q)≠ 0 do
            c ←POP(Q)
 5:         if DIAGNOSIS(c) then
                OUTPUTDIAGNOSIS(c)
                ENQUEUESIBLINGS(Q, c)
            else
                if (s ←NEXTBESTSTATE(H, c))⊭⊥ then
10:                 PUSH(Q, s)
                else
                    ENQUEUESIBLINGS(Q, c)
                end if
            end if
15:     end while
    end procedure
```

In its main loop, Algorithm 2 selects such a term c from the hierarchical node that the heuristic estimate $f(c)$ is maximized. When a consistent conjunction of terms is chosen from all the nodes in the hierarchy, OUTPUTDIAGNOSIS is invoked to send the result to the user.

The auxiliary functions PUSH, POP, and SIZE perform the respective priority queue manipulation on Q. The initial state in the search tree, returned by INITIALSTATE, is the empty term. The selection of next candidate states to be added to the search queue is done by the functions NEXTBESTSTATE and ENQUEUESIBLINGS. The former chooses the child state of the current state c and uses this term s from it, which again maximizes the utility function $f(s)$. When reaching a leaf or inconsistent node, ENQUEUESIBLINGS is used to queue the siblings of the current node and all its ancestors. Only consistent terms are added to the queue, and OUTPUTDIAGNOSIS should keep track of the diagnoses which are already discovered to prevent duplicate output. A practical approach is to keep the diagnoses in a trie [Forbus and de Kleer, 1993].

Let us illustrate the workings of Algorithm 2 on the hierarchical problem $HP = \langle H, OBS \rangle$, with a hierarchy H consisting of three nodes V_1, V_2, and V_3, two edges $e_1 = \{V_1, V_2\}$, $e_2 = \{V_1, V_3\}$ and a root node V_1. The DNF expressions in V_1, V_2, and V_3 are $\phi_1 = \{\}$, $\phi_2 = (h_1 \wedge \neg a \wedge b) \vee (h_1 \wedge a \wedge \neg b) \vee (\neg h_1)$ and $\phi_3 = (h_2 \wedge \neg b \wedge c) \vee (h_2 \wedge b \wedge \neg c) \vee (\neg h_2)$, respectively. We assume that $OBS = c$ and $P_i(h_i) = 0.95$ for $i = 1, 2$.

After instantiating the terms in V_1, V_2, and V_3 with *OBS* and removing the inconsistent terms we get the search tree in Figure 2.

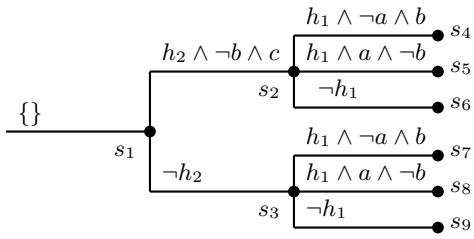

Figure 2: The search tree constructed by Algorithm 2 applied on an example hierarchical diagnosis problem.

Initially, the state s_1 is pushed on the queue. Its best child is s_2 as it has higher probability than s_3: $f(s_2) = 0.95$, while $f(s_3) = 0.05$. When s_2 is popped next, its first child s_4 is skipped as it leads to inconsistency (in the b literal) and s_5 is pushed on the Q. Next s_5 is retrieved (it is the only state on the queue) and it is a leaf node, hence its health variables form a diagnosis. After showing the diagnosis, Algorithm 2 pushes the best consistent siblings of all the predecessors of s_5 on the queue. These are s_6 and s_3. Now s_3 has the the highest probability in the queue, hence it is popped and its best child s_7 is pushed. In the next step s_7 is popped, which is a diagnosis. The process continues until, finally, we pop s_9 from the queue; a state which has the lowest probability diagnosis: $D = \{\neg h_1, \neg h_2\}$.

Normally, the performance of Algorithm 2 is not sensitive to the choice of the a-priori health probabilities for the components. This can change for observations leading to multiple faults, especially if these faults are in more than one subsystem, which is unlikely for well-engineered systems providing subsystem isolation. Algorithm 2 works faster for well-decomposed trees (i.e., having a small number of shared variables). As we will see in Section 4, this is not always the case. In practice, however, subsystems sharing more variables appear at the low levels of the hierarchy, constraining the number of solutions. Such subsystems are likely to be flattened by the pre-processing part of our hybrid algorithm thus leading to a fast overall diagnosis for systems well decomposed at the top level and constrained at the nodes, appearing close to the leaves of the tree.

Algorithm 2 differs from the traditional A* by the way it constructs its search tree. The depth of the search tree equals the number of subsystems and the order in which they are traversed depends on the hierarchy of the system (i.e., terms of top-level systems show near the root of the search-tree). In Section 4 we show the speedup of the algorithm as function of the compilation depth of the hierarchical DNF input, compared to traditional A*.

4 Experimental Results

In this section we empirically assess the performance of the algorithms using three test-cases: models of a 2-bit (FA2) and a 16-bit (FA16) full-adder, and a model of a Low Frequency Array radio-telescope (LOFAR). All our algorithms described in this paper are implemented as part of a toolkit based on the modeling language LYDIA (Language for sYstem DIAgnosis [van Gemund, 2003]).

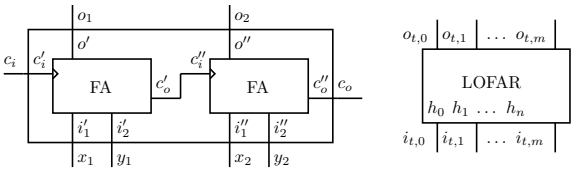

Figure 3: A 2-bit full-adder (left), composed of two full-adders. In one LOFAR element (right) t observations in time instantiate the same health variables and t different sets of observation variables.

The three benchmark systems are modeled in LYDIA which allows us to describe propositional hierarchical systems. The FA2 model is constructed of two full-adders, each full-adder constructed of two half-adders and an OR-gate. The half-adders consist of a XOR and an AND gate. The FA16 model demonstrates multi-level subsystem repetition. The top-level system consists of two 8-bit full-adders, each of them composed of two 4-bit full-adders, etc. (cf. Figure 3).

The LOFAR model describes the behavior of a single LOFAR element (the full system is designed to have 25 000 identical elements). The LOFAR system is hierarchical (i.e., it contains subsystems for an Analog to Digital Processing component, filters, selectors, etc.). The leaf components are described in terms of propositional expressions (e.g., defined are their inputs/outputs under loss of power, etc.).

Characteristic for all the models is that they are under-constrained at the top-level. The LOFAR model features a conjunction of 10 observations of the LOFAR input/output variables (cf. Figure 3), the flat DNF equivalent of a single element being constrained ($|\phi| < 10^4$). The 10 top-level instances of a LOFAR element have different sets of input and output variables but they share the same health variables, which allows reasoning about the state of the system by successively instantiating a new set of input variables. This leads to a space explosion in the top-level system if we would try to completely flatten it.

Table 1 summarizes some of the input characteristics of the hierarchical benchmark models, where $|E|$ denotes the number of edges in a hierarchy, $|V|$ denotes the number of nodes, and d denotes the depth of a hierarchy. Column N denotes the number of variables in a hierarchy and column M shows how many of them are health variables. The system with maximal number of variables in a hierarchy is $SD_{\max}$ and the number of variables in $SD_{\max}$ is denoted as $|SD_{\max}|$. The FA2 and FA16 models are weak-fault, while LOFAR is predominantly strong-fault.

All our tests are performed on a 333 MHz Ultra-SPARC 10 workstation with 256 MB of memory, running Solaris 8.

Model	N	M	$\|SD_{\max}\|$	d	$\|E\|$	$\|V\|$
FA2	50	20	4	4	7	6
FA16	283	160	4	7	13	9
LOFAR	387	54	7	7	65	34

Table 1: Size of the input models.

4.1 Hierarchical CNF to DNF Transformation

As mentioned earlier, our hybrid approach permits compilation to be limited for large models where full flattening (i.e., compilation to completely flat DNF) would be too explosive. In the experiments we consider three hierarchical depths (t), i.e., fully hierarchical (postfix H), partially flattened (postfix P), and fully flat (postfix F). The hierarchical depths of the models FA2/H, FA16/H, and LOFAR/H are shown in Table 1. We have chosen the (compilation) depths of FA2/P and LOFAR/P to be $t = 2$ and the depth of FA16/P to be $t = 4$. In this case, FA2/P contains full-adders in its leaf nodes, FA16/P is composed of flat 2-bit adders and LOFAR/P contains flat representations of a LOFAR element in its leaf nodes.

In all of three the models $|SD_{\max}|$ is small (cf. Table1). As a result, FA2, FA16 and LOFAR compile to DNF/H in 0.08 s, 0.19 s, and 0.3 s, respectively. It is visible that, for a well-decomposed system, the conversion from CNF/H to DNF/H consumes negligible time.

In order to compare the hierarchical compilation approach to the traditional (combinatorial) CNF to DNF conversion method, we need to convert the CNF/H representations to CNF/F (CNF/F is also needed for the flat A* which we will use as a comparison base in the run-time part of our experiments). The time for this (trivial) transformation is also negligible and the CNF/H representations of FA2, FA16, and LOFAR are converted to CNF/F in 0.06 s, 0.24 s, and 1.67 s, respectively.

	CNF/F to DNF/F		CNF/H to DNF/F		CNF/H to DNF/P	
	t [s]	size	t [s]	size	t [s]	size
FA2	22 824	2 774	41.54	3 365	0.32	515
FA16	T		M		42.53	26 927
LOFAR	T		M		50.12	29 361

Table 2: Compilation time and resulting DNF sizes of the FA2, FA16, and LOFAR models. Timeouts are denoted by T and memory exhaustion by M.

Table 2 compares the traditional, non-hierarchical approach for compilation (converting CNF/F to DNF/F in the first column) with Algorithm 1. For FA2 the speedup achieved by Algorithm 1 in producing DNF/F is 549. The flat DNF obtained in a hierarchical way is a slightly bigger than the one obtained using a conventional CNF to DNF translator. With the bigger FA16 and LOFAR models the traditional CNF to DNF compilation results in a time-out, while Algorithm 1 exhibits a memory exhaustion when trying to fully flatten them due to the excessive size of their DNF equivalents. Partial flattening proceeds as shown by the table, and results in considerable diagnosis speedup as will be discussed below.

4.2 Hierarchical A* Performance

In the following experiments we compare the hierarchical A*, described in Algorithm 2, with traditional A*. The results are shown in Table 3. For all the experiments, we have chosen observations that exclude the nominal (all healthy) behavior of the system as a diagnosis (e.g., for FA2 and FA16 we have 0 on all the inputs and 1 on the carry output, which corresponds to single, and multiple faults).

	CNF/F (A*)	DNF/H	DNF/P	DNF/F
FA2	500.6	636.7	2 389.1	1 161.4
FA16	0.0067	0.51	613.3	-
LOFAR	1.36	0.87	36 034	-

Table 3: Run-time diagnosis rates [diagnoses/s], averaged over an interval of 300 s.

Algorithm 2 is similar in speed or better than (traditional) A*. Due to the small size of FA2, the 27 percent speedup indicates a similar performance of the two approaches. The results with the FA16 model, however, show a speedup factor of 76 in favor if the hierarchical search. In the case of LOFAR, in contrast, the diagnosis rate of Algorithm 2 is 56 percent slower than that of the traditional A* method. The faster result for the traditional A* search comes for the high probability of the diagnoses. On the other hand LOFAR/H exposes a small size ($|H| = 1\,781$) and considerable depth ($d = 7$). In addition to that, Algorithm 2 produces in all the cases (DNF/H, DNF/P, and DNF/F) diagnoses with almost a constant rate and the total number of diagnoses is close to that of CNF/F.

As to be expected, partial flattening increases the performance of the hierarchical method. For FA2/P, FA16/P and LOFAR/P we observe speedups of 5, 91 542 and 26 497, respectively. This is explained by the following. First, the symbolic multiplication in Algorithm 1 leads to shorter terms (i.e., terms which do not contain all the health variables in *SD*). Algorithm 2 preserves these shorter terms, with some health variables being immaterial, as opposed to the full health variable instantiations of the traditional A*. If the number of all health variables is n and a solution term has m variables ($m < n$), this term is expanded to 2^{n-m} full health instantiations. A second reason for the improved performance is that during the compilation of the partially flattened model, many inconsistent states are eliminated. Therefore, a partial flattening to a moderate depth (more would lead to an explosion in the size of the DNF/P) results in an additional speedup of the hierarchical approach compared to the traditional A*.

5 Conclusion

In this paper we have described a two-step hierarchical method for computing diagnoses. The first, preprocessing step, transforms a hierarchical CNF model of the system to a fully or partially flattened DNF. In the second step, this hierarchical DNF is input to a hierarchical A* search for states consistent with the observation. The heuristic used for the hierarchical A* search is the a-priori probability of a state.

The improved performance of the hierarchical approach over the traditional CNF to DNF conversion and non-hierarchical A*, gained in both parts of the algorithm are intuitive and are empirically demonstrated. The hierarchical CNF to (flat) DNF compilation performs approximately 500 times better than a traditional DPLL-based CNF to DNF converter. As conversion to a fully flat DNF is possible only for small models, the partial flattening mode is typically used, where more preprocessing immediately translates to dramatic run-time speedups. Experiments, involving a 2-bit and 16-bit Full Adder and a model of the LOFAR system, demonstrate a factor of 2 - 91 542 speedup in diagnosis with partially flattened hierarchies, corresponding to a one-only, compile-time investment of a mere 50 seconds.

Future work includes using more benchmarks, in particular evaluating the performance of the search related to the model size, implementing a conflict-directed version of the hierarchical A* algorithm, and devising a method for an automatic determination of the partial flattening depth, providing an optimal time/space trade-off in diagnosis computation.

References

[Darwiche, 1998] Adnan Darwiche. Model-based diagnosis using structured system descriptions. *JAIR*, 8:165–222, 1998.

[Davis and Putnam, 1960] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *JACM*, 7(3):201–215, 1960.

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

[Forbus and de Kleer, 1993] Kenneth Forbus and Johan de Kleer. *Building Problem Solvers*. MIT Press, 1993.

[Liberatore, 1998] Paolo Liberatore. On the compilability of diagnosis, planning, reasoning about actions, belief revision, etc. In *Proc. KR'98*, pages 144–155, 1998.

[Mozetič, 1991] Igor Mozetič. Hierarchical model-based diagnosis. *JMMS*, 35(3):329–362, 1991.

[Provan, 2001] Gregory Provan. Hierarchical model-based diagnosis. In *Proc. DX'01*, 2001.

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

[Tseitin, 1983] Gregory Tseitin. On the complexity of proofs in propositional logics. In *Automation of Reasoning: Classical Papers in Computational Logic (1967–1970)*, volume 2. 1983.

[van Gemund, 2003] Arjan van Gemund. LYDIA version 1.1 tutorial. Technical Report PDS-2003-001, Delft University of Technology, 2003.

[Vatan, 2002] Farrokh 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 Robert Ragno. Conflict-directed A* and its role in model-based embedded systems. *JDAM*, 2004.