A FAULT ISOLATION ALGORITHM FOR THE CASE OF MULTIPLE FAULTS AND MULTIPLE FAULT TYPES

Mattias Nyberg

Department of Electrical Engineering, Linköping University,
SE-581 83 Linköping, Sweden
Phone: +46-13285714, Fax: +46-13282035,
Email: matny@isy.liu.se

Abstract: Given a number of thresholded residuals, an algorithm for finding the diagnoses, i.e. possible faults, is presented. The algorithm is based on ideas used in diagnosis algorithms from the field of AI. It is capable of handling the case of multiple faults and multiple fault-types per component. The number of multiple faults is exponential in the number of components. To handle this complexity problem, logical formulas are used to efficiently represent diagnoses. The formulas obtained can easily be used to derive the set of all diagnoses or the set of most probable diagnoses.

Keywords: fault isolation, multiple faults, structured residuals, minimal diagnoses

1. INTRODUCTION

The purpose of fault isolation is to identify the faulty component or components. Fault isolation has, in the field of FDI, commonly been performed using so called *structured residuals* (Gertler and Singer, 1990). This means that the fault isolation is based on a table such as

	f_1	f_2	f_3	f_4
r_1	1	0	0	1
r_2	1	1	0	0
r_3	0	1	1	0
r_4	0	1	0	0

(1)

Each row is associated with a thresholded residual. Further, each column is typically associated with one component, i.e. f_1 represents a fault in component c_1, f_2 a fault in component c_2 etc. The fault isolation is then performed by matching columns to the actual residual response. For example, if residuals r_1 and r_2 are above their thresholds, and r_3 and r_4 below, the conclusion is that a fault in component c_1 is present.

This simple approach has two problems. First, only single faults are considered. Second, only one fault type per component is considered. In real larger systems, it is often not sufficient to consider only single faults and only one type of fault per component. Thus it is important to handle also the case of multiple faults and multiple fault types per component.

It could be argued that multiple faults are less probable and therefore of less importance. However, a common case is that systems are not immediately repaired even though a fault has been detected and isolated. The reason is that many faults are not critical for continued operation. The consequence of this is that the number of faulty components increases over time, and when the system is finally repaired, the number of faulty components may be substantially larger than one.

The reason why multiple fault types should be considered is the following. Usually components can fail in several ways. For example, a sensor can be affected by bias-fault, gain-fault, short-cut, or open circuit. For repairing the system or to take appropriate fault accommodation actions, it may be critical that the fault isolation not only localizes the faulty component but also concludes which the fault type is. Further on, some residuals may respond to a certain fault type but not to another, and it would be a waste not to utilize this additional information when doing fault isolation.

To solve the first problem, handling of multiple faults, one column for each possible multiple fault can be added to the isolation table. This approach was suggested in (Gertler, 1998). However, if N is the number of components, the number of columns needed is 2^N-1, which makes this approach computationally intractable for anything but small systems.

The second problem, multiple fault types per component, can be addressed by adding separate columns for each fault type, see (Nyberg, 2002). When combining this with a consideration of multiple faults, and letting M denote the number of fault types per component, the number of columns needed becomes $(M+1)^N-1$. Thus the computational burden is increased even further.

To perform fault isolation fast is critical in real-time systems. The reason is that fault isolation often is the basis for taking decision about fault accommodation. Knowing exactly the faulty component and type of the fault can make the difference between shutting down the process or continuing operation with only slightly degraded performance. If the fault isolation is too slow, the absence of fault isolation information in combination with for example safety requirements, may force the process to be shut down soon after that the fault is detected.

The requirements of handling multiple faults and multiple fault types per component, in combination with a requirement to perform fault isolation quickly, role out the approaches used in (Gertler, 1998; Nyberg, 2002). The contribution in the present paper is a new approach

that we will show to be significantly faster than the approaches in (Gertler, 1998; Nyberg, 2002). This new approach is based on a previous algorithm used in the field of AI (deKleer and Williams, 1987). The difference is that although the algorithm in (deKleer and Williams, 1987) can handle multiple faults in an efficient way, it can not handle the case of multiple fault types.

Before continuing the discussion, we need to formulate the "fault isolation problem" more precisely. Out of several possibilities, we aim at solving the following two problems. The first is to, given a set of thresholded residuals that have responded to some unknown fault or faults, compute the set of all *diagnoses*, i.e. all fault combinations that are consistent with the residual response. For large systems the number of such diagnoses can be quite large. Therefore the second problem considered is to compute the set of only so called *preferred diagnoses* (Dressler and Struss, 1992). The preferred diagnoses are, in one sense, all most probable diagnoses, and they have the advantage that they are considerably fewer than the set of all diagnoses.

Fault isolation in the case of several fault types has indeed been discussed earlier in the field of AI, see e.g. (Struss and Dressler, 1989; deKleer and Williams, 1989). However, these works aim at solving the whole diagnosis problem, and not only the fault isolation problem. That is, these works also include so called conflict recognition, which here corresponds to residual generation. Also, the model class they consider is models that can be handled by a so called ATMS (deKleer and Williams, 1987). Nevertheless, their basic ideas of how to solve the fault isolation problem could be compared to the algorithm discussed in the present paper. However, even though only the fault isolation part is considered, both these works admit complexity problems. To handle the complexity, (deKleer and Williams, 1989) searches only some of the most probable diagnoses, but this may also cause complexity problems in some cases. In (Struss and Dressler, 1989), only single-fault diagnoses are computed.

The paper is organized as follows. First some basic principles used are described in Section 2. The tool for our work is a logical framework, defined in Section 3. In Section 4, we generalize and formalize the notion of thresholded residual. Section 5 then describes the basic algorithm for solving the fault isolation problem. In Section 6 it is described how this algorithm can be used to solve the two problems of finding all diagnoses and the preferred diagnoses respectively. The approach is then illustrated on an application example in Section 7. Finally, an empirical comparison between the new approach and the column matching approach from (Gertler, 1998) is presented in Section 8.

2. BASIC PRINCIPLES

Before approaching the problem of constructing an efficient algorithm for fault isolation, we will replace the 1:s in the table (1) with X:s. The reason for this is that in real systems with noise, model uncertainties, and nonlinearities present, it is not possible to guarantee that a residual exceeds its threshold. So for example in the table (1), when f_2 is the present fault, only the residual r_3 may exceed its threshold. When using 1:s, the isolation result becomes incorrectly that f_3 is present. If instead the 1:s are replaced with X:s meaning that a residual only *may* respond to the failure, the isolation result would be that either f_2 *or* f_3 is present. Note that this is a correct conclusion. This issue has earlier been discussed in (Nyberg, 1999; Cordier *et al.*, 2000).

We will assume that the residual response to multiple faults can be derived from the single fault response. That is, the X:s in a multiple-fault column is chosen as the "union" of the X:s in the corresponding single fault columns. This assumption should hold in most real applications. By using only X:s and this assumption, the multiple-fault version of (1) becomes

	f_1	f_2	f_3	f_4	f_1f_2	f_1f_3	f_1f_4	$\dots f_1f_2f_3f_4$
r_1	X	0	0	X	X	X	X	$\dots$ X
r_2	X	X	0	0	X	X	X	$\dots$ X
r_3	0	X	X	0	X	X	0	$\dots$ X
r_4	0	X	0	0	X	0	0	$\dots$ X

(2)

Now instead of only a column matching, we will in this paper use a more sophisticated approach to search for diagnoses. To illustrate the principle, assume that only r_2 has responded. We see in the table (2) that the single faults f_1 and f_2 are diagnoses. By construction of the table, we also know directly that all multiple faults including f_1 or f_2, e.g. f_1f_2 and f_2f_3, are the remaining diagnoses. Then if also r_3 responds, the diagnoses are updated. The single fault f_1 can no longer be a diagnosis, but instead all multiple faults including f_1f_3 become diagnoses. The old diagnosis f_2, and all multiple diagnoses including f_2, are still diagnoses.

It will be shown that this principle of doing fault isolation is much more efficient than the use of column matching. It follows the approach used by diagnosis algorithms such as (Reiter, 1987; deKleer and Williams, 1987), developed within the field of AI. Note however that the algorithms in (Reiter, 1987; deKleer and Williams, 1987) cannot, in contrast to the new approach presented in the present paper, handle multiple fault types.

3. LOGICAL FRAMEWORK

This section presents a logical framework in which diagnosis conclusions can be represented in an efficient way. In the system to be diagnosed, identify a set of *components* that we want to diagnose. Each component is assumed to be in exactly one out of several *behavioral modes* which can be thought of as the fault status of the component. A behavioral mode can be for example no-fault, abbreviated NF, gain-fault G, bias B, open circuit OC, short circuit SC, unknown fault UF, or just faulty F. For our purposes, each component is abstracted to a variable specifying the mode of that component. Let $\mathcal{C}$ denote the set of such variables. For each component variable φ let $\mathbf{R}_\varphi$ denote the *domain* of possible behavioral modes, i.e. $\varphi \in \mathbf{R}_\varphi$.

We will now define a set of formulas to be used to express that certain components are in certain behavioral modes. If φ is a component variable in the set $\mathcal{C}$ and $M \subseteq \mathbb{R}_\varphi$, the expression $\varphi \in M$ is a formula. For example, if p is a pressure sensor, the formula $p \in \{NF, G, UF\}$ means that the pressure sensor is in mode NF, G, or UF. If M is a singleton, e.g. $M = \{NF\}$, we will sometimes write also $p = NF$. Further, the constant $\perp$ with value *false*, is a formula. If ϕ and γ are formulas then $\phi \wedge \gamma$, $\phi \vee \gamma$, and $\neg\phi$ are formulas.

A *system behavioral mode* is a conjunction containing a unique assignment of all components in $\mathcal{C}$. For example if $\mathcal{C} = \{p_1, p_2, p_3\}$, a system behavioral mode could be $p_1 = UF \wedge p_2 = B \wedge p_3 = NF$. Now note that by using

the logical framework, certain sets of system behavioral modes can be represented in an efficient way. For example, consider a single conjunction $D_1 = c_1 \in M_1 \wedge c_2 \in M_2 \wedge c_3 \in M_3$. Note first that to store this conjunction in the computer memory, we need only to store the three sets M_1, M_2, and M_3 which are usually small. If the system contains 7 components, D_1 is able to represent a set of $|M_1||M_2||M_3||\mathbf{R}_{c_4}||\mathbf{R}_{c_5}||\mathbf{R}_{c_6}||\mathbf{R}_{c_7}|$ system behavioral modes. If each M_i would have 3 elements, and each $\mathbf{R}_{c_i}$ 5 elements, the single conjunction D_1 would represent 16875 system behavioral modes.

In accordance with the theory of first order logic we say that a formula ϕ is a semantic consequence of another formula γ, and write $\gamma \models \phi$, if all assignments of the variables $\mathcal{C}$ that make γ true also make ϕ true. This can be generalized to sets of formulas, i.e. $\{\gamma_1, \ldots, \gamma_n\} \models \{\phi_1, \ldots, \phi_m\}$ if and only if $\gamma_1 \wedge \cdots \wedge \gamma_n \models \phi_1 \wedge \cdots \wedge \phi_m$. If it holds that $\Gamma \models \Phi$ and $\Phi \models \Gamma$, where Φ and Γ are formulas or sets of formulas, Φ and Γ are said to be equivalent and we write $\Gamma \simeq \Phi$.

4. DIAGNOSTIC TESTS

In the introduction we discussed a diagnosis system based on thresholded residuals. Instead of thresholded residuals we will use the generalized notion of *diagnostic tests*. A diagnostic test is considered to be any device that takes measured or known signals as inputs and gives as output a conclusion about the behavioral modes. Thus a diagnostic test may for example include a thresholded residual, an approach involving parameter estimation, or some statistical signal processing.

A diagnostic test is a special case of a classical binary hypothesis test (Casella and Berger, 1990). To each test there is a *null hypothesis* H^0 which states the assumption that is tested by the diagnostic test. In this paper we assume that H^0 is expressed in the logical framework defined in Section 3. For example, consider a test based on a thresholding of residual r_3 in the table (2). Assuming that f_i means a fault of a corresponding component c_i where $\mathbf{R}_{c_i} = \{NF, F\}$, the null hypothesis would be $H^0 = c_2 \in \{NF\} \wedge c_3 \in \{NF\}$. In general a diagnostic test usually tests some relation between measured and known signals in the form of a so called *parity relation* or *analytical redundancy relation*. The null hypothesis is then the conjunction of all behavioral mode assumptions that had to be made to derive this consistency relation. More details on the relation between a consistency relation and its corresponding null hypothesis can be found in (Nyberg and Krysander, 2003).

The fact that the diagnostic tests are assumed to be binary means that there are two possible outcomes when testing the null hypothesis. Either it is rejected which means that the conclusion drawn from the test is the alternative hypothesis $H^1 = \neg H^0$. Otherwise the null hypothesis is not rejected which means that no conclusion is drawn from the test. This "asymmetry" of the test corresponds to the use of only X in the isolation table, as exemplified by (2). The underlying reason is that for example noise forces us to draw no conclusion when H^0 is not rejected, in favour of being able to draw the conclusion H^1 when H^0 is rejected. For the example of r_3 in (2), this means that when the residual is below its threshold, no conclusion is drawn, and when the residual is above the threshold, we draw the conclusion $H^1 = \neg H^0 \simeq c_2 \in \{F\} \vee c_3 \in \{F\}$.

Following this principle means that, without loss of generality, we can assume that all alternative hypotheses are written as

$$H^1_k = \varphi_1 \in M_1^C \vee \varphi_2 \in M_2^C \vee \cdots \vee \varphi_n \in M_n^C \quad (3)$$

where the φ_i:s are placeholders for component variables from the set $\mathcal{C}$, $\varphi_j \not\equiv \varphi_k$ if $j \neq k$, and $\emptyset \neq M_i \subset \mathbf{R}_{\varphi_i}$. Note that not all variables in $\mathcal{C}$ are necessarily contained in the formula (3).

5. THE BASIC FAULT-ISOLATION ALGORITHM

The task of the algorithm presented in this section is to, given a set of test conclusions, compute an expression representing all diagnoses. Before describing the algorithm, we first give a formal definition of the term diagnosis.

A diagnosis is a system behavioral mode consistent with all test conclusions. That is, if $\mathbb{P}$ is the set of all test conclusions, a system behavioral mode d is a *diagnosis* if $\{d\} \cup \mathbb{P} \not\models \perp$ or equivalently $d \models \mathbb{P}$.

The algorithm is based on two principles to make it efficient. The first is to, instead of working with sets of diagnoses as in the column matching approach, diagnoses are represented using the logical framework presented in Section 3. Then our problem of computing an expression representing all diagnosis could simply be solved by taking the conjunction of all test conclusions. However, as stated in Section 1, our final goal is to compute explicit sets of all diagnoses and preferred diagnoses, and the conjunction of all test conclusions does not bring us closer to this final goal. Therefore the algorithm is based also on a second principle, namely to represent diagnoses using a specific normal form, which is defined in the next section. When using this normal form, the operations to compute all or preferred diagnoses become almost trivial, as will be described in Section 6. An additional advantage with this normal form is that it avoids redundancy in the representation.

5.1 *A Normal Form*

When presenting the normal form, we will use the notation D_i for the i:th conjunction $(\varphi_{i1} \in M_{i1} \wedge \varphi_{i2} \in M_{i2} \wedge \cdots \wedge \varphi_{in_i} \in M_{in_i})$. The φ_{ij}:s are placeholders for component variables from the set $\mathcal{C}$, and φ_{ij} represents the j:th variable in the i:th conjunction. We will now say that a formula is in *maximal normal form* MNF if it is written on the form

$$(\varphi_{11} \in M_{11} \wedge \varphi_{12} \in M_{12} \wedge \cdots \wedge \varphi_{1n_1} \in M_{1n_1}) \vee \ldots$$
$$\cdots \vee (\varphi_{m1} \in M_{m1} \wedge \cdots \wedge \varphi_{mn_m} \in M_{mn_m})$$

where $\varphi_{ij} \not\equiv \varphi_{ik}$ if $j \neq k$, and

a) no conjunction is a consequence of another conjunction, i.e. for each conjunction D_i, there is no conjunction D_j, $j \neq i$, such that $D_i \models D_j$, and
b) each M_{ij} is a nonempty proper subset of $\mathbf{R}_{\varphi_{ij}}$, i.e. $\emptyset \neq M_{ij} \subset \mathbf{R}_{\varphi_{ij}}$.

Note that the definition says, that for example φ_{11} and φ_{21} can represent the very same variable, but φ_{11} and φ_{12} cannot. Note also that the requirements (a) and (b) guarantee that a formula is compact in the sense that it does not contain redundant conjunctions and a that each conjunction does not contain redundant assignments.

For an example consider the following two formulas containing pressure sensors p_1, p_2, and p_3, where all have the behavioral modes $\mathbb{R}_{p_i} = \{NF, G, B, UF\}$.

$$p_1 \in \{UF\} \wedge p_2 \in \{B, UF\} \vee p_3 \in \{UF\}$$
$$p_1 \in \{UF\} \wedge p_2 \in \{B, UF\} \vee p_1 \in \{G, UF\}$$

The first formula is in MNF but not the second since $p_1 \in \{UF\} \wedge p_2 \in \{B, UF\} \models p_1 \in \{G, UF\}$.

5.2 *The Algorithm*

As said above, the task of the algorithm is to compute a formula representing all diagnoses. This is done by computing a new formula $\mathcal{Q}$ in MNF such that $\mathcal{Q} \simeq \mathbb{P}$. The algorithm consists of a main algorithm and a subroutine conj$(\mathcal{D}, \mathcal{P})$.

The inputs to the subroutine are an MNF-formula $\mathcal{D} = \bigvee_i D_i$ and a test conclusion $\mathcal{P} = \bigvee_j P_j$. The purpose of the subroutine is then to compute the output $\mathcal{Q}$ in MNF with the property $\mathcal{Q} \simeq \mathcal{D} \wedge \mathcal{P}$. Before presenting the details of the subroutine, we present the main algorithm.

Algorithm 1. Main Algorithm
Input: a set $\mathbb{P}$ of test conclusions
Output: $\mathcal{Q}$
remove a test result $\mathcal{P}$ from $\mathbb{P}$ and let $\mathcal{Q} := \mathcal{P}$
forall $\mathcal{P} \in \mathbb{P}$ do
    $\mathcal{Q} :=$ conj$(\mathcal{Q}, \mathcal{P})$
next

Theorem 1. If the test conclusions in $\mathbb{P}$ are on the form (3), then the output $\mathcal{Q}$ from Algorithm 1 is in MNF and $\mathcal{Q} \simeq \mathbb{P}$.

The proof of this theorem can be found in (Nyberg, 2006).

To describe the principles of the subroutine, consider the following expansion of $\mathcal{Q}$:

$$\mathcal{Q} \simeq \mathcal{D} \wedge \mathcal{P} = \bigvee_i D_i \wedge \bigvee_n P_n \simeq \tag{4a}$$

$$\simeq D_1 \wedge P_1 \vee D_1 \wedge P_2 \vee D_1 \wedge P_3 \vee \ldots \cdots \vee D_2 \wedge P_1 \vee D_2 \wedge P_2 \vee \ldots \tag{4b}$$

Clearly, (4b) is equivalent to $\mathcal{Q}$ but not necessarily in MNF. The algorithm below computes the expansion (4b) but, when used together with Algorithm 1, avoids to include those redundant conjunctions that destroy the MNF property.

Algorithm 2. Subroutine conj$(\mathcal{D}, \mathcal{P})$
Input: a formula $\mathcal{D}$ in MNF, a test conclusion $\mathcal{P}$
Output: $\mathcal{Q}$
$\mathcal{D}_{old} = \mathcal{D}$
forall $D_i \in \mathcal{D}$ do
    if $D_i \not\models \mathcal{P}$ then
        Remove D_i from $\mathcal{D}_{old}$
        forall $P_j \in \mathcal{P}$ do
            Let D_{new} be a conjunction in MNF such that $D_{new} \simeq D_i \wedge P_j$
            forall $D_k \in \mathcal{D}$, $k \neq i$ do
                if $D_{new} \models D_k$ then
                    goto LABEL1
                endif
            next
            $\mathcal{D}_{add} := \mathcal{D}_{add} \vee D_{new}$
            LABEL1
        next
    endif
next
$\mathcal{Q} := \mathcal{D}_{old} \vee \mathcal{D}_{add}$

To illustrate the subroutine, consider the following small example where $\mathcal{C} = \{p_1, p_2, p_3\}$ and the domain of behavioral modes for each component is $\mathbb{R}_{p_i} = \{NF, G, B, UF\}$.

$$\mathcal{D} = D_1 \vee D_2 = p_1 \in \{G, B, UF\} \vee p_3 \in \{G, UF\}$$
$$\mathcal{P} = P_1 \vee P_2 = p_2 \in \{B, UF\} \vee p_3 \in \{G, B, UF\}$$

First the condition $D_1 \not\models \mathcal{P}$ is fulfilled which means that D_1 is removed from $\mathcal{D}_{old}$ and the inner loop is entered. There a D_{new} is created such that $D_{new} \simeq D_1 \wedge P_1 = p_1 \in \{G, B, UF\} \wedge p_2 \in \{B, UF\}$. This D_{new} is then compared to D_2 when checking the condition $D_{new} \models D_2$. The condition is not fulfilled which means that D_{new} is added to $\mathcal{D}_{add}$. Next a D_{new} is created such that $D_{new} \simeq D_1 \wedge P_2 = p_1 \in \{G, B, UF\} \wedge p_3 \in \{G, B, UF\}$. Also this time the condition $D_{new} \models D_2$ is not fulfilled, implying that D_{new} is added to $\mathcal{D}_{add}$. Next, the conjunction D_2 is investigated but since $D_2 \models \mathcal{P}$ holds, D_2 is not removed from $\mathcal{D}_{old}$ and the inner loop is not entered. The algorithm output is finally formed as

$$\mathcal{Q} := \mathcal{D}_{old} \vee \mathcal{D}_{add} = D_2 \vee (D_1 \wedge P_1 \vee D_1 \wedge P_2) = = p_3 \in \{G, UF\} \vee p_1 \in \{G, B, UF\} \wedge p_2 \in \{B, UF\} \vee \vee p_1 \in \{G, B, UF\} \wedge p_3 \in \{G, B, UF\} \tag{5}$$

It can be verified that $\mathcal{Q} \simeq \mathcal{D} \wedge \mathcal{P}$. Also, it can be seen that $\mathcal{Q}$ is in MNF.

5.3 *Details of the Subroutine*

To implement the algorithm of the subroutine, some more details need to be known. The first is how to check the condition $D_i \models \mathcal{P}$. To illustrate this, consider an example where $D_i = c_1 \in M_1^D \wedge c_2 \in M_2^D \wedge c_3 \in M_2^D$ and $\mathcal{P} = c_2 \in M_2^P \vee c_3 \in M_3^P \vee c_4 \in M_4^P$. We realize that the condition $D_i \models \mathcal{P}$ holds if and only if $M_2^D \subseteq M_2^P$ or $M_3^D \subseteq M_3^P$. Thus, this example shows that in general, $D_i \models \mathcal{P}$ holds if and only if D_i and $\mathcal{P}$ contain at least one common component c_i where $M_i^D \subseteq M_i^P$.

The second detail is how to find an expression Q_{new} in MNF such that $Q_{new} \simeq D_i \wedge P_j$. To illustrate this, consider an example where $D_i = c_1 \in M_1^D \wedge c_2 \in M_2^D$ and $P_j = c_2 \in M_2^P$. Then Q_{new} will be formed as $D_{new} = c_1 \in M_1^D \wedge c_2 \in M_2^D \cap M_2^P$. This means that $D_{new} \simeq D_i \wedge P_j$ and that Q_{new} will be in MNF.

The third detail is how to check the condition $D_{new} \models D_k$. It can be realized that $D_{new} \models D_k$ holds if and only if (1) D_k contains only components that are also contained in D_{new}, and (2) for all components c_i contained in both D_{new} and D_k, i.e. $c_i \in M_i^n$ and $c_i \in M_i^D$ respectively, it holds that $M_i^n \subseteq M_i^D$.

6. USING ALGORITHM 1 FOR FAULT ISOLATION

After having processed all test conclusions using Algorithm 1, we obtain an MNF-expression $\mathcal{Q}$ representing all diagnoses. Given this $\mathcal{Q}$, this section describes how to compute the sets of all diagnoses and preferred diagnoses.

To compute the set of all diagnoses is the simplest. First, the sets of diagnoses represented by each conjunction in $\mathcal{Q}$ are derived. The set of all diagnoses is then obtained by taking the union of the sets derived from each conjunction.

Note that when the number of components is large, the number of diagnoses represented by $\mathcal{Q}$ is typically a very large number. It may in this case not be feasible

to compute all diagnoses. Therefore we describe next an alternative solution, namely to prioritize among the diagnoses and pick out only the "most desirable" ones.

6.1 *Minimal/Preferred Diagnoses*

In the literature, several criteria for defining what is meant by "most desirable" diagnoses have been used. Probably the most commonly used is *minimal* or *preferred diagnoses*. The concept of minimal diagnoses was originally proposed in (Reiter, 1987; deKleer and Williams, 1987) for systems where each component has only two possible behavioral modes, i.e. the no-fault mode and a faulty mode. In (Dressler and Struss, 1992) the concept was generalized to components with any number of modes. They also used the word *preferred diagnoses* instead of minimal diagnoses.

The basic idea is that the behavioral modes for each component are ordered in a partial order defining that some behavioral modes are more desirable, or preferred, than other. For example, NF is usually preferred over any other mode, and a simple electrical fault, such as short-cut or open circuit, is usually preferred over other more complex behavioral modes. Further, an unknown fault UF may be the least preferred mode.

For a formal definition let $b_c^1 \geq_c b_c^2$ denote the fact that for component c, the behavioral mode b_c^1 is equally or more preferred than b_c^2. For each component, this relation forms a partial order on the behavioral modes. Further, these relations induce a partial order on the system behavioral modes. Let d_1 and d_2 be two system behavioral modes $d_i = \wedge_{c \in \mathcal{C}}(c = b_c^i)$. Then we write $d_1 \geq d_2$ if for all $c \in \mathcal{C}$ it holds that $b_c^1 \geq_c b_c^2$. A preferred diagnosis can then formally be defined as a diagnosis d such that there is no other diagnosis d' where $d' > d$.

Under some assumptions, it can be shown that a preferred diagnosis is a most probable diagnosis given the set of all test conclusions $\mathbb{P}$. This justifies the concept of preferred, or minimal, diagnoses. Let $P(c = b_c)$ be the prior probability that component c is in behavioral mode b_c. To know the exact value of the prior $P(c = b_c)$ may be very difficult or even impossible. Therefore assume only that for each component, the priors are partially ordered and let this define the relation $\geq_c$. Then under the assumption that faults occur independently of each other, the preferred diagnoses, will be exactly the most probable ones given the test conclusions, i.e. the diagnoses d such that there is no other diagnosis d' where $P(d'|\mathbb{P})$ is known to be larger than $P(d|\mathbb{P})$.

6.2 *Computing Preferred Diagnoses*

Given an MNF-expression $\mathcal{Q}$ in MNF, we will now discuss how to obtain the set of preferred diagnoses. For each conjunction in $\mathcal{Q}$, find the preferred diagnoses consistent with that conjunction, and collect all diagnoses found in a set Δ. In the general case, two conjunctions may contain preferred diagnosis d_1 and d_2 respectively and it may hold that $d_1 > d_2$. Therefore the set Δ can contain non-preferred diagnoses. These can be removed by a simple pairwise comparison and the remaining ones will be the set of preferred diagnoses.

For an example, consider two components c_1 and c_2 where $\mathbf{R}_{c_i} = \{NF, E, F\}$ and $NF > E > F$, and a third component c_3 where $\mathbf{R}_{c_i} = \{NF, B, G\}$ with the only relations $NF > B$ and $NF > G$. Then consider

$$\begin{aligned}\mathcal{Q} =& c_1 \in \{E\} \wedge c_3 \in \{B, G\} \vee \\ & c_1 \in \{E, F\} \wedge c_2 \in \{E, F\} \wedge c_3 \in \{B, G\}\end{aligned}$$

The preferred diagnoses consistent with the first conjunction are $c_1 = E \wedge c_2 = NF \wedge c_3 = B$ and $c_1 = E \wedge c_2 = NF \wedge c_3 = G$. The preferred diagnoses consistent with the second are $c_1 = E \wedge c_2 = E \wedge c_3 = B$ and $c_1 = E \wedge c_2 = E \wedge c_3 = G$, but these will be removed from Δ.

7. APPLICATION EXAMPLE

As an application example we will study an electrical driver for the fuel injectors of a 6-cylinder automotive engine. This system has six components, namely one driver for each of the six injectors. Each driver has eight behavioral modes: NF, SBB (short between banks), SC (stuck closed), SCG (short circuit to ground), SLB (short circuit on low side to ground), OL (open load), SHB (short circuit on high side to battery), and UF. The complexity of this example is illustrated by the fact that in total, there are $8^6 = 262144$ system behavioral modes. Thus, an isolation table, as the one shown in (2) including all multiple faults, would have 262143 columns.

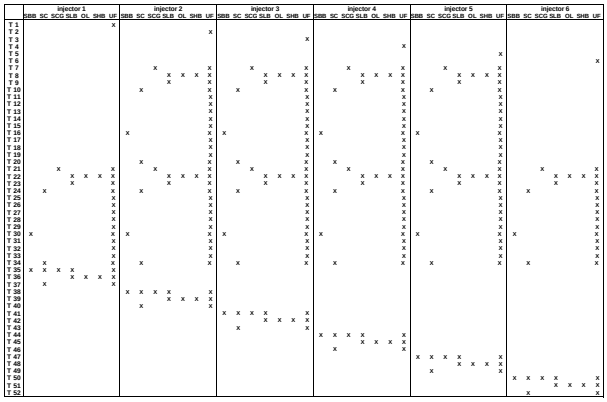

Fig. 1. The isolation table for the electrical-driver system, shown for single faults.

There are 52 diagnostic tests monitoring this system. Their response to different single faults are shown in the isolation table in Figure 1. This table thereby specifies the null hypothesis for each diagnostic test as described in Section 2.

For the example, we now assume that tests 10, 30, 38, 44, and 45 have responded to some fault. Then the set of all diagnoses and the set of all preferred diagnoses are to be computed. For comparison we use both the column matching approach from (Gertler, 1998), but extended to multiple fault types as described in Section 1, and the approach based on Algorithm 1, both implemented in SciLab. When calculating preferred diagnoses, we use a partial order defined by the relations $NF < b$ for all behavioral modes $b \neq NF$ and $b < UF$ for all $b \neq UF$. Using these approaches, the total number of diagnoses is computed to be 31960. Further, the number of preferred diagnoses is 27. Two examples of preferred diagnoses are $\langle NF, SBB, NF, UF, NF, NF \rangle$ and $\langle NF, SC, SBB, SLB, NF, NF \rangle$.

For comparison, a variant of the column matching approach to compute preferred diagnoses, was also implemented. The principle used was that the isolation table is traversed from left to the right. The system behavioral mode b of each column is compared to a set Ω of already

computed preferred diagnoses, and if concluded that $b < d$ for some diagnosis $d \in \Omega$, then b is neglected, and otherwise added to Ω if the test response matches the column. Furthermore, if concluded that $d < b$, d is removed from Ω.

The computation time needed for both approaches is shown below. For comparison, also the time needed for Algorithm 1 to compute the MNF-formula $\mathcal{Q}$ representing all diagnoses is shown.

	All diagnoses	Preferred diagnoses	MNF formula
column matching	3144s	8198s	NA
new algorithm	11.5s	11.4s	10.7s

We can note that the new approach, based on Algorithm 1, computes all diagnoses 273 times faster than the column matching approach. Further, the new approach computes preferred diagnoses 719 times faster than the column matching approach. Additionally, it is seen that for the new approach, the extra time needed to compute all or preferred diagnoses from the MNF formula, is less than 10% of the time needed to compute only the MNF formula.

8. COMPARISON OF COMPUTATION TIMES

For further comparison between the column matching approach and the new approach, a number of 132 test cases were randomly generated. The test cases represent systems with between 4 and 7 components, where each component has 4 possible behavioral modes. The number of diagnostic tests that respond varies between 2 and 12.

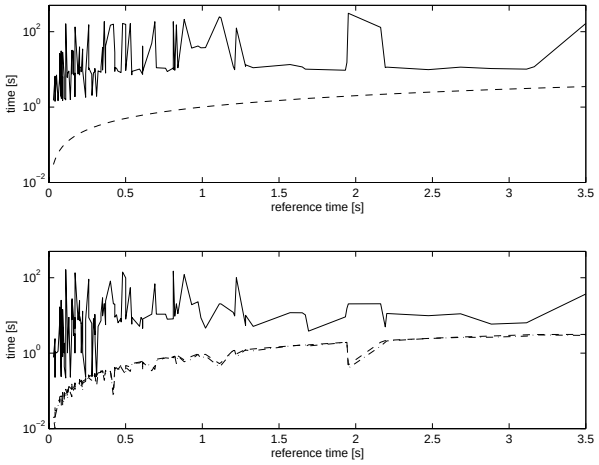

Fig. 2. Execution times for the column matching approach (solid line) and the new approach (dashed line).

In Figure 2, the results for the 132 test cases are shown. The upper plot shows the computation time when computing all diagnoses. The lower plot shows the computation time when computing preferred diagnoses. In both plots, the reference time on the x-axis is chosen to be the computation time for the new approach when computing all diagnoses. In the lower plot, also the time needed by Algorithm 1 to compute the MNF formula is shown as a dashed-dotted line.

Even though the test cases studied represent quite small systems (maximum 7 components), it is seen that the performance for the new approach is considerably better than the column matching approach. In some cases the performance is more than 1000 times better. This holds true both when calculating all diagnoses and preferred diagnoses. Lastly, one can note that the extra time needed to compute all diagnoses or preferred diagnoses from the MNF formula, is almost negligible compared to the time needed to compute the MNF formula.

9. CONCLUSIONS

In this paper a fault isolation algorithm capable of handling the case of multiple faults and multiple fault modes per component has been presented. Compared to earlier approaches (Gertler, 1998; Nyberg, 2002; Struss and Dressler, 1989; deKleer and Williams, 1989) we are able to compute all diagnoses, and to control the complexity, by using an efficient representation of the diagnoses. The representation is efficient since diagnoses need not to be enumerated explicitly. A single conjunction D_i in an MNF-formula is potentially able to represent a large number of diagnoses. To prioritize among the possibly large set of diagnoses, it was shown how minimal or preferred diagnoses could be extracted from an MNF-formula representing all diagnoses.

In a comparative study, including both a real application and a large number of randomly generated test cases, it was shown that this new approach outperforms the more traditional column matching approach in (Gertler, 1998).

10. REFERENCES

Casella, G. and R.L. Berger (1990). *Statistical Inference*. Duxbury Press.

Cordier, M.-O., P. Dague, M. Dumas, F. Lévy, J. Montmain, M. Staroswiecki and L. Travé-Massuyès (2000). Ai and automatic control theory approaches of model-based diagnosis : links and underlying hypotheses. SAFEPROCESS. IFAC. Budapest, Hungary. pp. 274–279.

deKleer, J. and B.C. Williams (1987). Diagnosing multiple faults. *Artificial Intelligence* **Issue 1, Volume 32**, pp. 97–130.

deKleer, J. and B.C. Williams (1989). Diagnosis with behavioral modes. IJCAI. pp. 1324–1330.

Dressler, O. and P. Struss (1992). Back to defaults: Characterizing and computing diagnoses as coherent assumption sets. ECAI. pp. 719–723.

Gertler, J. (1998). *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker.

Gertler, J. and D. Singer (1990). A new structural framework for parity equation-based failure detecation and isolation. *Automatica* **26**(2), 381–388.

Nyberg, Mattias (1999). Automatic design of diagnosis systems with application to an automotive engine. *Control Engineering Practice* **Issue 8, Volume 7**, pp. 993–1005.

Nyberg, Mattias (2002). Model-based diagnosis of an automotive engine using several types of fault models. *IEEE Transaction on Control Systems Technology* **10**(5), 679–689.

Nyberg, Mattias (2006). A generalization of a minimal-hitting set algorithm to handle behavioral modes. 17th Int. Workshop on Principles of Diagnosis. Penaranda de Duero, Burgos, Spain.

Nyberg, Mattias and Mattias Krysander (2003). Combining AI, FDI, and statistical hypothesis-testing in a framework for diagnosis. In: *Proceedings of IFAC Safeprocess'03*. Washington, USA.

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

Struss, P. and O. Dressler (1989). 'physical negation' - integrating fault models into the general diagnosis engine. IJCAI. pp. 1318–1323.