Model-Based Diagnosis with Multi-Label Classification

Betty Keren, Meir Kalech, and Lior Rokach
Information Systems Department, Ben-Gorion University of the Negev, Israel
betty.keren1@gmail.com,{kalech,liorrk}@bgu.ac.il

ABSTRACT

Model-Based Diagnosis (MBD) is one of the leading artificial intelligence approaches that copes with the diagnosis problem. MBD is known as a hard problem and grows exponentially in the size of the system. In this paper, we propose a novel approach that combines MBD with multi-label classification. We propose to build a classifier that maps symptoms of the system to possible faults. The major advantage of this approach is by reducing significantly the online computational complexity; The learning process of the relations between the observation and the diagnosis is performed in advance offline, and then online, by using the classifier, we can immediately return the diagnosis. This paper addresses several challenges: 1) modeling the MBD problem as a classification problem, 2) generating informative samples for the training set, 3) verifying sound and minimal diagnosis.

1 INTRODUCTION

With the development of technology over the years and the overgrowing use of large scale and complex systems, the problem of tracking and diagnosing faulty components in the system became more difficult. Thus, the need for automated diagnosis is increased. Automated diagnosis is concerned with reasoning about the health of systems, including the identification of abnormal behavior, the isolation of faulty components and the prediction of system behavior under normal and abnormal conditions.

Model-Based Diagnosis (MBD) is one of the leading artificial intelligence approaches that copes with the diagnosis problem (de Kleer and Williams, 1987). In MBD approach, a model of the system is being built. The diagnoser observes the actual behavior of the system and predicts its behavior by the model. Discrepancies between the observation and the prediction—symptoms—are used as the input for the diagnosis algorithms which produce a set of possible faults that can explain such symptoms.

MBD is known as a hard problem and grows exponentially in the size of the system - i.e. the number of components in the system. In case of large scale systems using MDB is impractical due to its computational complexity. Unfortunately, the fact that many of the systems today (e.g. network systems, robots etc.) have a huge number of components, leads to the infeasibility of MBD approach in those systems.

Many of the previous diagnosis methods propose deterministic reasoning and search algorithms to address the complexity challenge (de Kleer and Williams, 1987; Williams and Ragno, 2007). These methods guarantee sound diagnoses; however, in large systems they fail, due to the infeasible runtime. Feldman et al. (Feldman *et al.*, 2010b) propose a stochastic algorithm, SAFARI. Although this method does not guarantee minimal diagnoses, it presents solutions which are close to minimal cardinality in a very low runtime.

In this paper, we propose to combine MBD with multi-label classification (Tsoumakas *et al.*, 2010). Conventional classification tasks deal with problems where each item should be assigned to exactly one category from a finite set of available labels. This type of classification is referred to in the literature as "single-label". Conversely, in multi-label classification, an instance can be associated with several labels simultaneously. Multi-labeling is a very common problem in text classification: medical documents, Web pages, and scientific papers, for example, often belong simultaneously to a number of concept classes. Due to its increasing practical relevance as well as its theoretical interest, multi-label classification has received more attention from the machine learning community in recent years and many recent studies look for efficient and accurate algorithms for coping with this classification challenge.

For the MBD problem, we propose to build a classifier that maps symptoms of the system to possible faults. In particular, the inputs and the outputs use as the attributes for the learning algorithm while the faulty components indicate the different classes. The major advantage of this approach is by reducing significantly the online computational complexity; The learning process of the relations between the observation and the diagnosis is performed in

advance offline, and then online, by using the classifier, we can immediately return the diagnosis.

To implement this process we go through the next steps: 1) We create a training set of samples by simulating fault injections and registering the relation between the inputs+outputs and the faulty components. Since the fault space and the input space are exponential we randomly choose the samples. One of the major challenges that we address in this paper is how to select the samples. Then 2) we train a classifier with the training set offline, and finally at runtime 3) we enter an observation (inputs+outputs) to the classifier and get the diagnosis candidate. Since this candidate is not guaranteed to be sound, 4) we verify its soundness and try to minimize it by stochastic local search.

An hybrid approach of MBD and machine learning has been investigated in previous work (Niggemann *et al.*, 2009; Alonso-González *et al.*, 2010). This paper addresses several interesting challenges related to multiple faults:

1. How to train the classifier? in particular, how to choose the inputs (attributes) and the faulty components (classes)?
2. What classification approach is appropriate to the diagnosis problem?
3. How to verify sound and minimal diagnoses at runtime?

Preliminary experiments in the DXC framework (Second International Diagnostic Competition (DXC 10), 2010) on the $74xxx$ benchmark systems, show the performance of the multi-label classification approach.

2 RELATED WORK

Many of the previous diagnosis methods propose deterministic reasoning and search algorithms. One approach proposes to divide the diagnosis process to two stages, first to find the conflict sets which each one of them includes at least one fault, and then to generate multiple faults that explain the observation by hitting set over the conflicts (de Kleer and Williams, 1987; Williams and Ragno, 2007). These methods guarantee sound diagnoses, and some of them are even complete; however, in large systems they tend to fail, due to the infeasible runtime or space.

Other compilation based approaches try to represent the system in other formulations in order to cope with the complexity. For instance, Torta and Torasso represent the MBD problem with OBDD (Torta and Torasso, 2004), while Huang and Darwiche represent it with DNNF (Huang and Darwiche, 2005). There are some attempts to solve the diagnosis problem with SAT solvers (Feldman *et al.*, 2006), however all the above approaches are deterministic and since the diagnosis problem is hard they do not scale well.

There are other nondeterministic approaches. For instance, Feldman et al. (Feldman *et al.*, 2010b) propose a stochastic algorithm, SAFARI, in which a candidate is verified by a series of trials where each trial tries to improve the former one. Although this method does not guarantee minimal diagnoses, it presents solutions which are close to minimal cardinality in a very low runtime.

There is much work which considers the diagnosis problem in terms of inductive learning. These approaches try to learn the relations between the symptoms and the faults (Murray *et al.*, 2006). One of the disadvantages of most of these approaches is that they learn only a single fault rather than multiple faults (Balakrishnan and Honavar, 1998). We, on the other hand, propose a method to learn multiple faults with multi-label classification.

There are some works which propose a hybrid approach that combines model-based diagnosis and classification. Stein at al. (Niggemann *et al.*, 2009) present diagnosis for automotive applications which uses a model of the vehicle systems to train a classifier in compilation time with machine learning techniques based on linear regression and decision trees. Alonso et al. (Alonso-González *et al.*, 2010) also use AdaBoost approach to train an ensemble of classifiers in temporal systems. In this approach, they perform fault detection and localization based on consistency, and fault identification with the time series classifiers. Both papers focus on strong models, where the components are represented with behavior modes. In our paper we focus on large-scale weak model systems. In such systems it is possible to train the classier only on a very small subset of the sample space. In addition, since we address multiple faults which is a basic requirement in MBD, we propose multi-label classification which has not been combined before with model-based diagnosis.

3 MODEL DESCRIPTION

In this section we will define first the classification problem and the multi-label classification (section 3.1). Then we will show how to implement the MBD problem in terms of multi-label classification (section 3.2). In section 3.3 we will describe how to select the samples for the training set ,and in section 3.4 we will show how to train the classifier. Finally, in section 3.5 we will describe how to get a diagnosis at runtime and verify its consistency.

3.1 Multi-Label Classification

The classification problem can be stated as follows: given training data $\{(x_1, y_1), ..., (x_n, y_n)\}$ produce a classifier h, such that $h(x)$ can be evaluated for any possible value of x as close as possible to the true group label y. We use the term "attributes" for x and "classes" for y. For the training data-set, the true labels y_i are known but will not necessarily match their in-sample approximations. For new observations, the true labels y_j are unknown, and then it is a prime target for the classification procedure to classify the observation correctly.

In multi-label classification, an instance can be associated with several labels simultaneously. The multi-label classification (MLC) problem can be described as follows: Each instance x_i may be classified with a subset of labels Y_i, such that $Y_i \subseteq L$. Where L is a given set of predefined binary labels $L = \{\lambda_1, ..., \lambda_n\}$. For a given set of labeled examples $D = \{x_0, x_1, ..., x_m\}$ the goal of the learning process is to find a classifier $h : X \rightarrow Y$, which

maps an object $x \in X$ to a set of its classification labels Y, such that $h(x) = \{\lambda_1, ..., \lambda_n\}$ for all $x \in X$. The main feature distinguishing multi-label classification from a regular classification task is that a number of labels have to be predicted simultaneously.

3.2 MBD as a Multi-Label Classification Problem

The MBD problem is defined by three sets; 1) a set of assumables (components) ($COMPS$), 2) a set of first-order sentences which represent a model description of the behavior of the components and the connections between them (SD), and 3) observations which are represented by a set of first-order sentences (OBS). A diagnosis problem arises when these sets are inconsistent with each other.

Definition 1. Diagnosis Problem. *Given* $\{SD, COMPS, OBS\}$ *the diagnosis problem (DP) arises when*

$$SD \cup \{\neg AB(COMPS_i) | COMPS_i \in COMPS\} \cup OBS \vdash \bot$$

where $AB(c)$ *is an unary predicate which is true when component* c *is faulty.*

Once there is inconsistency, the diagnosis algorithm tries to find a set of faulty components which leads to consistency.

Definition 2. *A* diagnosis *is a set* $\Delta \subseteq COMPS$ *such that:*

$$SD \cup \{AB(COMPS_i) | COMPS_i \in \Delta\} \cup$$
$$\{\neg AB(COMPS_i) | COMPS_i \in COMPS - \Delta\} \cup OBS \nvdash \bot$$

We are interested in *minimal diagnosis*, meaning, no proper subset of it is a diagnosis.

Superficially, it seems that there is no relation between MBD and classification, since typically a classification approach is used for problems where there are many samples. The samples use for training a classier: a model of how the attributes affect the classes. On the other hand, MBD does not use samples at all but a predefined model. However, a deep vision shows that it is possible to train a classifier based on a given model in MBD, by generating samples with simulated faulty samples.

In particular, we propose to represent the observation by the attributes (x) and the faulty components by the classes (y). In this way we can learn the relations between the observations and the faults. We assume in that idea that there are connections between the observations and the faulty components. Obviously, this assumption is correct in many real world systems like mechanical systems, software etc. For instance, a car veering may caused due to a faulty left wheel. Learning the relations in electric circuits is challenging, since this is a many-to-many relationship, and so there could be many explanations for each observation and vice versa.

MBD approach does not assume to have samples for the system, and so we propose to actively generate samples to train a classifier. A sample is composed of: 1) the attributes, in our case the inputs (IN) and outputs (OUT), and 2) the classes, in our case multiple faulty components. We use a multi-label classification since multiple faults can explain the observation. The process of generating the training set (TS) is described in algorithm 1. We iteratively run through some constant N to build N samples (line 2). For each sample we randomly set inputs and assumables (lines 3–4) and propagate through the system to explore the outputs (line 5). The inputs+outputs are used as the attributes and the assumables as the class (lines 6–7).

Algorithm 1 TRAINING_SET
(**input**: system description SD
output: a training set TS)

```
1:  TS ← ∅
2:  for all i ∈ {1, ..., N} do
3:      IN ← set inputs' values
4:      COMPS ← set assumables
5:      OUT ← propagate through SD
6:      attribute ← IN ⋃ OUT
7:      classes ← COMPS
8:      sample ← ⟨attribute, class⟩
9:      TS ← TS ⋃ {sample}
10: end for
11: return TS
```

To clarify the propagation process through the system (line 5), we should define first a weak fault model:

Definition 3. weak fault model (WFM). *In weak fault model (WFM), the assumables are defined for the components by the predicate* $AB(COMPS_i)$. *The system description is equivalent to* $\neg AB(COMPS_1) \Rightarrow F_1 \wedge, ..., \wedge \neg AB(COMPS_n) \Rightarrow F_n$, *where* $F_1, ..., F_n$ *are propositional formulas.*

In WFM, we can propagate through the behavior of all the healthy components, but not through those components that are abnormal, since the system description does not determine their behavior. To propagate through the behavior of faulty components we need to define a strong fault model (Struss and Dressier, 1989).

Definition 4. strong fault model (SFM). *In strong fault model (SFM), the system description is equivalent to* $\neg AB(COMPS_1) \Rightarrow F_{1,1} \wedge AB(COMPS_1) \Rightarrow F_{1,2} \wedge, ..., \wedge \neg AB(COMPS_n) \Rightarrow F_{n,1} \wedge AB(COMPS_n) \Rightarrow F_{n,2}$, *where* $F_{1,1}, F_{1,2} ..., F_{n,1} F_{n,2}$ *are propositional formulas.*

In SFM, we can propagate and get the correct behavior of a component as well as its faulty behavior. For simplicity, we define in this paper that a faulty behavior returns the negation of the correct behavior (if $\neg AB(COMPS_i) \Rightarrow F_i$ then $AB(COMPS_i) \Rightarrow \neg F_i$). Thus we build a training set by generating samples in a strong model as described. For instance, Table 1 presents several samples created for the full adder in Figure 1. The sequences in the first column represent the values of the three inputs and two outputs (A, B, C, D, E), and the components appear in the second column, represent the multi-label faulty components. For instance, propagating through the full adder with the inputs $A = 1, B = 1, C = 1$ and the faulty component X_1 (such that it returns the negative value of xor) produces the output $D = 0, E = 1$. Therefore,

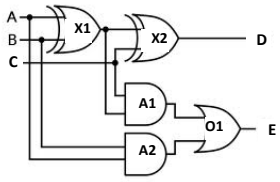

Figure 1: A full adder.

Attributes	Classes
$1, 1, 1, 0, 1$	X_1
$1, 1, 1, 1, 1$	X_1, X_2
$1, 1, 1, 0, 1$	X_2, A_1
$1, 0, 0, 0, 0$	X_1
$1, 0, 0, 1, 0$	X_1, X_2
$1, 0, 0, 0, 1$	X_2, A_1

Table 1: A set of samples for the full adder in Figure 1.

the attributes are $1, 1, 1, 0, 1$ and the multi-label is X_1.

There are still two challenges in this approach, first how to choose the inputs for each sample? second, how to choose the faulty components? the first challenge will be addressed in future work, meanwhile we randomly select inputs, the second challenge will be addressed in the next section.

3.3 Building the Training Set

Given m the number of inputs and n the number of components, the sample space is 2^{nm} (the product of 2^m inputs and the power set of the faulty components). Unfortunately, this space is feasible only for very small systems. We should choose only a very small subset of this space, and the challenge in this section is how to choose the most informative samples?

It is easy to train all single fault components since this number is linear in the size of the system and so we actually train all of them. But double fault and more must be partially trained. We will describe our approach on the double fault selection and then extend it to multiple fault.

An intuitive approach could be to train the components that are more likely to fail. Under the assumption in MBD, that faulty components are independent, we could take the most likely pair of components. However, we assume in our algorithm a uniform distribution of failure, and thus we cannot prefer any pair.

We propose a new approach, with no prior probabilities, to learn the probability of pairs to be in the same diagnosis. For this we present first a conflict, as defined by Reiter (Reiter, 1987):

Definition 5. conflict. *A conflict for* $\{SD, COMPS, OBS\}$ *is a set* $CONF \subseteq COMPS$ *such that:*

$$SD \cup \{\neg AB(COMPS_i) | COMPS_i \in CONF\} \cup OBS$$

is inconsistent.

A conflict is a set of assumables that their healthy assumption leads to inconsistency, thus at least one of the assumables is faulty. If the conflict set is minimal the observation can be explained by each one of the faulty assumable. That means that statistically, two assumables that appear in the same conflict are more likely to emit **the same** observation. A diagnosis will include a representative of each conflict; that is the hitting set (de Kleer and Williams, 1987; Reiter, 1987). Thus two assumables that appear in the same diagnosis are more likely to emit **different** observations. Obviously, there are counter examples for each one of these claims, but based on the definitions of conflict and observation these claims are statistically straightforward.

In our approach we try to learn the dissimilarity between the outputs of faulty components. The more difference between components' output the more likely they are in the same diagnosis. Algorithm 2 describes how to build the dissimilarity matrix between components. We iteratively run through Z random inputs (line 4). For each input we simulate a faulty behavior of each component separately (lines 6–7) and propagate the inputs through the system to emit the outputs (line 8). For each output we define a different matrix (OUT^i) in which we save its values for the different inputs (lines) over the faulty components (columns) (lines 9–11). The element OUT^i_{jk} represents the value of output OUT_i with the inputs IN_j when $COMP_k$ is faulty. Then, in lines 14–19, we compute the dissimilarity between every pair of components regarding every output (the dissimilarity between every two columns in OUT^i). The dissimilarity is the ratio between the number of differences and the number of inputs. We save the dissimilarity in the matrix SIM^i for each output and normalize each matrix separately (line 20). Finally, we sum up the dissimilarity matrices and normalize into NOR_SIM (lines 22–25). The element NOR_SIM_{jk} represents the dissimilarity between the components $COMP_j$ and $COMPS_k$. The more dissimilarity between components the more likely they will be in the same diagnosis.

To generate a training set of double fault components, we randomly select pairs of faulty components based on the dissimilarity matrix. We propagate random inputs through the system with the faulty components and register the outputs. A sample in the training set is an instance of the inputs+outputs and the faulty components. Let us demonstrate the whole process based on the circuit in Figure 1. Assume three random inputs: $IN^1 = \{A = 1, B = 1, C = 1\}$, $IN^2 = \{A = 1, B = 1, C = 0\}$ and $IN^3 = \{A = 1, B = 0, C = 0\}$. The OUT matrices for the outputs D and E are presented in Tables 2 and 3. For instance, OUT^D_{11} represents the upper-left cell which is the output of a scenario where the input presented in the first row (IN_1) and by simulating X_1 as faulty. The corresponding SIM matrices are presented in Tables 4 and 5. The values in parentheses are before normalizing. The diagonal line is 0 since it represents the dissimilarity between a component to itself. The values in the elements below the diagonal are the same as the values in the upper side and so we ignore them. For instance, the value $\frac{1}{3}$ in the second element of the first line in Table 5 represents the dis-

Algorithm 2 SIMILARITY_MATRIX
(**input**: system description SD
input: components $COMPS$
output: dissimilarity matrix NOR_SIM)

$OUT^1, ..., OUT^{|OUT|}$
$SIM^1, ..., SIM^{|OUT|}$
$NOR\_SIM \leftarrow \{\{0\}\}$
for all $i \in \{1, ..., Z\}$ **do**
  $IN \leftarrow$ set random inputs' values
  for all $j \in \{1, ..., n\}$ **do**
    set $\bigwedge_{k \neq j} \neg AB(COMP_k) \wedge AB(COMP_j)$
    $OUT \leftarrow$ propagate through SD
    for all $f \in \{1, ..., |OUT|\}$ **do**
      $OUT^f_{ij} \leftarrow OUT_f$
    end for
  end for
end for
for all $f \in \{1, ..., |OUT|\}$ **do**
  for all $i \in \{1, ..., n\}$ **do**
    for all $j \in \{i, ..., n\}$ **do**
      $SIM^f_{ij} \leftarrow$ similarity between column OUT^f_i and OUT^f_j
    end for
  end for
  normalize SIM^f
end for
for all $f \in \{1, ..., |OUT|\}$ **do**
  $NOR\_SIM \leftarrow NOR\_SIM + SIM^f$
end for
normalize NOR_SIM
return NOR_SIM

#input	X_1	X_2	A_1	A_2	O_1
1, 1, 1	0	0	1	1	1
1, 1, 0	1	1	0	0	0
1, 0, 0	0	0	1	1	1

Table 2: The OUT^D matrix for output D in Figure 1.

similarity between the first two columns in Table 3, that indicates that these vectors are different in one value out of three (the normalized value is $\frac{1}{18}$). Finally, Table 6 is a normalized summation over Table 4 and 5. According to this matrix, it is very likely to select for the training set the pair $\{X_1, O, 1\}$ and it is unlikely at all to choose $\{A_2, O_1\}$. Note that indeed the conflict sets for the above observations reflect the similarities. For instance, the components $\{X_1, X_2\}$ appear together almost in all conflicts that one of them appears in; indeed the similarity between them is very high $\frac{35}{36}$ (the complement of the dissimilarity).

Extending the training set to triple fault and more is performed in the same manner based on the dissimilarity matrix computed for pairs. For each dissimilarity between $COMPS_i$ and $COMPS_j$, we compute the extension of adding $COMPS_k$, by adding the dissimilarity probabilities of $COMPS_i$ with $COMPS_k$ and of $COMPS_j$ with $COMPS_k$. As will be shown in the next section, the extension will be actually done only for a subset of the space.

3.4 Classification Methodology

Given the set of samples we can train a classifier. We adopt the AdaBoost meta-algorithm for the classification process (Freund and Schapire, 1995). In this method, a set of s classifiers are built sequentially. On each round, the weights of each incorrectly classified example are increased (or alternatively, the weights of each correctly classified example are decreased), so that the new classifier on the next round will focus on those examples. Finally, the whole s classifiers use for the classification task by averaging their results.

#input	X_1	X_2	A_1	A_2	O_1
1, 1, 1	1	1	1	0	0
1, 1, 0	1	0	1	0	0
1, 0, 0	0	0	1	1	1

Table 3: The OUT^E matrix for output E in Figure 1.

$COMP_i$	X_1	X_2	A_1	A_2	O_1
X_1	0	0	$\frac{1}{6}(1)$	$\frac{1}{6}(1)$	$\frac{1}{6}(1)$
X_2	-	0	$\frac{1}{6}(1)$	$\frac{1}{6}(1)$	$\frac{1}{6}(1)$
A_1	-	-	0	0	0
A_2	-	-	-	0	0
O_1	-	-	-	-	0

Table 4: The SIM matrix for output D in Figure 1.

We adapt AdaBoost for the MBD problem, by training every new classifier to an increased set of diagnoses. In this way we train the classifiers to cope with different sizes of diagnosis. Specifically, in the first round we train the classifier with the training set as built in Section 3.3. The classification algorithm in each round could be any multi-label classification, in our case we used $RAKEL$ (Tsoumakas and Vlahavas, 2007) with the classification algorithm $J48$. Then, in each round we update the training set to train a new classifier as follow:

1. Each sample in the training set is tested in the classifier of the previous round and its false negative rate is measured. The false negative rate is the ratio between the number of misclassified components and the injected faulty components (Feldman *et al.*, 2010a).
2. Every sample is associated with a weight: $w_i = a + b * (false\ negative\ rate)$, where a and b are constants that affirm $a, b \in [0, 1]\ and\ a + b \leq 1$. Note that the lower false negative rate the higher w_i, since we encourage to retrain misclassified samples rather than samples that have been classified correctly.
3. To build the new training set for the next classifier, we randomly choose z samples from the previous training set considering the weights of the samples (with repeats). Each chosen sample is increased by one component. The new component is extended by the initial dissimilarity matrix, as described in the previous section. In this way, we give more chance to misclassified samples on the one hand, and on the other hand we train diagnoses in larger sizes.
4. Finally, we train a new classifier with the new training set using $RAKEL$.

Eventually we trained s classifiers, each classifier for different size of diagnosis. At runtime, we use all the classifiers for the diagnosis process. Each classifier returns multi-label which represents

$COMP_i$	X_1	X_2	A_1	A_2	O_1
X_1	0	$\frac{1}{18}(\frac{1}{3})$	$\frac{1}{18}(\frac{1}{3})$	$\frac{1}{6}(1)$	$\frac{1}{6}(1)$
X_2	-	0	$\frac{1}{9}(\frac{2}{3})$	$\frac{1}{9}(\frac{2}{3})$	$\frac{1}{9}(\frac{2}{3})$
A_1	-	-	0	$\frac{1}{9}(\frac{2}{3})$	$\frac{1}{9}(\frac{2}{3})$
A_2	-	-	-	0	0
O_1	-	-	-	-	0

Table 5: The SIM matrix for output E in Figure 1.

$COMP_i$	X_1	X_2	A_1	A_2	O_1
X_1	0	$\frac{1}{36}$	$\frac{1}{9}$	$\frac{1}{6}$	$\frac{1}{6}$
X_2	-	0	$\frac{5}{36}$	$\frac{5}{36}$	$\frac{5}{36}$
A_1	-	-	0	$\frac{1}{18}$	$\frac{1}{18}$
A_2	-	-	-	0	0
O_1	-	-	-	-	0

Table 6: The normalized summation matrix of matrices 4 and 5.

a set of components (denoted by $COMPS_\Delta \subseteq COMPS$), where each one of them is associated with a confidence value in a range of $[0, 1]$ (the confidence represents the likelihood of the component to be classified as a diagnosis). Let $\langle COMP_i, conf_i \rangle$ denote the confidence $conf_i \in [0, 1]$ of component $COMP_i \in COMPS_\Delta$. To determine the actual confidence of a component based on the whole classifiers, we evaluate the weight of each classifier and return the weighted average confidence of the component over the classifiers.

We will demonstrate this process in the next example. Assume a training set of pairs as presented in Table 7. After training a classifier on this training set we check each one of the samples on this classier. The third column presents the results of the classifier on these samples, the fourth column presents the false negative rate of each sample and the fifth column presents the weight, where $a = 0.1$ and $b = 0.9$. Based on the weights we create another z samples, and each one of them is extended by a new component. The extension is done with Table 6. For instance, to determine whether to extend the pair X_1, O_1 by either X_2, A_1 or A_2, we add each one of them to X_1 and O_1 separately and normalize as presented in Table 8. We train a new classifier with the new training set.

3.5 Consistent Diagnosis

At runtime, the AdaBoost obtains the observations (inputs and outputs) and returns a set of components with their confidence value. We set two constants, P for the threshold of the confidence, and Q for the number of components in the diagnosis. To select a diagnosis candidate, we randomly choose Q components with a confidence greater than P, out of the components returned by AdaBoost. $COMPS_{PQ} \subseteq COMPS_\Delta$ represents the chosen diagnosis. This diagnosis is not guaranteed to be minimal nor consistent, and thus we should check these properties.

Algorithm 3 describes the process of the diagnosis verification. The obtained diagnosis candidate is checked through DPLL for consistency. If it is consistent then we try to minimize it to get a minimal diagnosis (lines 6–10). Similarly to SAFARI (Feldman *et al.*, 2010b), our algorithm tries to minimize the diagnosis by iteratively flipping in each round

Attributes	Classes	Actual classification	False negative	Weight
$1, 1, 1, 0, 0$	X_1, O_1	X_1	0.5	0.55
$1, 1, 1, 1, 1$	X_1, X_2	X_1, X_2	0	0.1
$1, 1, 1, 0, 1$	X_2, A_1	X_2, A_1	0	0.1
$1, 0, 0, 0, 1$	X_1, O_1	X_1, O_1, A_2	0	0.1
$1, 0, 0, 1, 0$	X_1, X_2	O_1	1	1
$1, 0, 0, 0, 1$	X_2, A_1	A_1	0.5	0.55

Table 7: A set of double fault samples and their weights.

Triple fault	Dissimilarity
X_1, O_1, X_2	$\frac{1}{6} + \frac{1}{36} + \frac{5}{36} = \frac{1}{3}$
X_1, O_1, A_1	$\frac{1}{6} + \frac{1}{9} + \frac{1}{18} = \frac{1}{3}$
X_1, O_1, A_2	$\frac{1}{6} + \frac{1}{6} + 0 = \frac{1}{3}$

Table 8: Dissimilarity of $\langle X_1, O_1, ? \rangle$.

the value of one abnormal assumable in the diagnosis, until we obtain an inconsistent diagnosis. The advantage of our algorithm is that the next flipped assumable is not randomly selected as in SAFARI, but it is determined based on the confidence value of the components returned by the classifier (line 8).

If the diagnosis candidate obtained by AdaBoost is inconsistent, then it must be a subset of another diagnosis, thus we repeatedly increase the diagnosis by adding more components (lines 2–5). The selection of the next component is done by a stochastic local search (Hoos and Sttzle, 2004), considering again the confidence value of the components. The returned diagnosis is the first one to pass the consistency check by DPLL.

Note that the consistency check through DPLL and the minimality check are relevant only to WFM (see Definition 3), which is in the interest of this paper. Although, the learning stage, described in Section 3.2, was done under the assumption of SFM, which takes into consideration that abnormal components behave in negative way. However, since a diagnosis in SFM must be a diagnosis in WFM (Console and Torasso, 1991), by injecting diagnosis samples in SFM, we actually add diagnosis samples that are valid in WFM too. Thus we can infer, by the classifier, diagnoses that can be checked with DPLL.

4 EVALUATION

There are many constants that should be addressed in our method, for instance, how many inputs are

Algorithm 3 CHECK_CONSISTENCY
(**input**: diagnosis $COMPS_{PQ}$
output: a diagnosis set Δ)

1: $\Delta \leftarrow \emptyset$
2: **while** $COMPS_{PQ}$ is inconsistent (check with DPLL) **do**
3: $COMP_i \in COMPS_\Delta \backslash COMPS_{PQ} | \forall j, conf_i \geq conf_j$
4: $COMPS_{PQ} \leftarrow COMPS_{PQ} \cup \{COMP_i\}$
5: **end while**
6: **while** $COMPS_{PQ}$ is consistent (check with DPLL) **do**
7: $\Delta \leftarrow COMPS_{PQ}$
8: $COMP_i \in COMPS_{PQ} | \forall j, conf_i \leq conf_j$
9: $COMPS_{PQ} \leftarrow COMPS_{PQ} \backslash \{COMP_i\}$
10: **end while**
11: return Δ

System	Description	$\|OBS\|$	$\|COMPS\|$
74182	4-bit carry-lookahead generator	14	19
74L85	4-bit magnitude comparator	14	33
74283	4-bit adder	14	36
74181	4-bit ALU	22	65

Table 9: An overview of the 74XXX benchmark circuits.

System	Random	Multi-Label Classification
74182	13.432	10.07
74L85	21.79	17.54
74283	25.66	21.08
74181	46.86	41.16

Table 10: Consistency checks until verifying a diagnosis.

required for the dissimilarity matrix? how many samples are required in each iteration of AdaBoost? how to choose the inputs? how to set Q and P? etc. Since multi-label classification for MBD has not been done before, we do not have answers for these challenges, and so in this section we just present some preliminary results that show that our approach is significant. We plan to further examine all the above open questions in order to present a consistent and robust approach.

We run preliminary experiments on the $74xxx$ benchmark systems (presented in Table 9), to check some of the attributes of the multi-label classification approach. We run the scenarios of the DXC-2010 (Second International Diagnostic Competition (DXC 10), 2010). Note that these scenarios include also probes. In the first experiment we show the benefit of our approach over a random approach. We have built the dissimilarity matrix by simulating each single fault component through Z=100 input samples (Section 3.3). We have trained the above systems through a small set of samples: We have trained all single faults, and quarter of the double faults, and the same number for three, four and five multiple faults. Each diagnosis was trained through 100 inputs. For instance, in system 74263, only 9 diagnoses were trained in each round of AdaBoost (for each diagnosis size 1–5), with 100 inputs.

At runtime, we set $Q = 1$ and $P = 0.5$. In fact, we literately added one component in every iteration, in the order of their confidence, until we got a consistent diagnosis (by checking it through DPLL). We compared it with a random approach, in which we added in each iteration a random component (rather than the component with the next confidence value). We compared the number of added components until a consistent diagnosis was verified. Table 10 summarizes the average number of components in each approach. As shown, our multi-label classification approach requires less consistency checks to find a diagnosis than a random approach. Note that the large number of components in the diagnosis is mainly due to the probes in the scenarios; these probes significantly reduce the number of consistent states. To summarize, this experiment shows that even learning over a very small sample set improves the process of finding a diagnosis. We believe that increasing the sample set will further improve the diagnosis. We will examine this issue in the future.

The next experiment examines the utility of the diagnosis as defined in (Feldman *et al.*, 2010a). The utility is defined as follows: $1 - \frac{n(N+1)}{f(n+1)} - \frac{\overline{n}\overline{N}+1}{f(\overline{n})+1}$ where n is the false negative, $\overline{n}$ is the false positive, and N is the the set of healthy components from the viewpoint of the diagnoser. For this experiment, we ask to check how the confidence affects the utility. The utility of a scenario is a measurement that takes into consideration the tradeoff between the false negative and false positive. In our algorithm we determine the diagnosis based on the confidence value returned by the classifier for each one of the components. Typically the classification is determined as a result of the confidence, for instance, the multi-label diagnosis contains all the components with confidence greater than 0.5. Obviously, it is not guaranteed to be consistent, but we are interested, in the next experiment, to evaluate the false negative and false positive as a function of the confidence threshold.

To this aim, we checked the utility for different values of confidence threshold 0.1, 0.2, 0.3, 0.4, 0.5. We compared the utility to that of a random selection algorithm. We set the random algorithm to select the same number of components as the multi-label selected with the confidence threshold. Figures 2, 3, 4 and 5 summary the results for the four benchmarks. The x-axis is the confidence threshold and the y-axis is the utility. In each one of these graphs we can see that the utility of our approach outperforms the utility of a random approach. In addition we can see that in big systems the utility reduces with the threshold. The reason is that the number of components grows and thus the probability of finding the exact faulty components reduces.

To summarize, we can see that the classification increases the utility of the diagnosis over random selection. In addition, the size of the system and the confidence are correlated and affect the decision of what threshold is the best one. In the future we plan to investigate this issue.

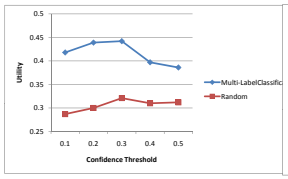

Figure 2: Utility for 74182.

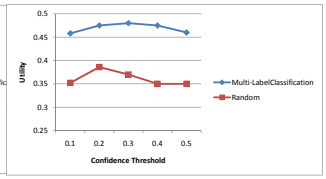

Figure 3: Utility for 74L85.

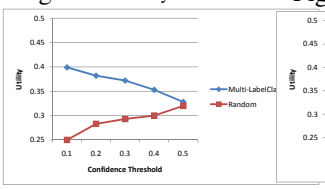

Figure 4: Utility for 74283.

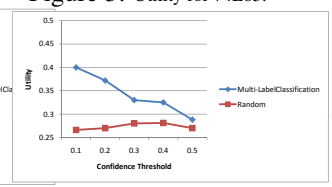

Figure 5: Utility for 74181.

5 SUMMARY AND FUTURE WORK

In this paper we presented a model-based diagnosis engine using multi-label classification. In this ap-

proach the observation uses as the attributes and the faulty components in the diagnosis are the classes. We try to learn the relations between the observations and the diagnoses. For this, we proposed to simulate a training set by generating samples of the most likely multiple fault diagnoses. We adapted the AdaBoost approach for our MBD problem, and used RAKEL as multi-label classifier with $J48$ learning algorithm. Preliminary results show the advantages of our approach over a random approach.

In the future, we propose to investigate and learn the constants in our method. In particular, we want to learn the weights of the classifier in AdaBoost, the number of required samples as a function of the size of the system, the number of samples to learn the dissimilarity matrix and the ideal number of classifiers in AdaBoost. In addition, in this paper we chose the inputs randomly. We want to investigate a better approach to chose the inputs by orthogonal arrays.

REFERENCES

(Alonso-González *et al.*, 2010) Carlos J. Alonso-González, Juan José Rodríguez, Óscar J. Prieto, and Belarmino Pulido. Ensemble methods and model based diagnosis using possible conflicts and system decomposition. In *Proceedings of the 23rd international conference on Industrial engineering and other applications of applied intelligent systems - Volume Part II*, IEA/AIE'10, pages 116–125, Berlin, Heidelberg, 2010. Springer-Verlag.

(Balakrishnan and Honavar, 1998) Karthik Balakrishnan and Vasant Honavar. Intelligent diagnosis systems. *Journal of Intelligent Systems*, 8(3/4):239–290, 1998.

(Console and Torasso, 1991) Luca Console and Pietro Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7(3):133–141, 1991.

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

(Feldman *et al.*, 2006) Alexander Feldman, Jurryt Pietersma, and Arjan van Gemund. A multi-valued SAT-based algorithm for faster model-based diagnosis. In *Proceedings of the Seventeenth International Workshop on Principles of Diagnosis (DX'06), Peñaranda de Duero, Burgos, Spain*, June 2006.

(Feldman *et al.*, 2010a) A. Feldman, T. Kurtoglu, S. Narasimhan, S. Poll, D. Garcia, Johan de Kleer, Lukas Kuhn, and A.J.C. van Gemund. Empirical evaluation of diagnostic algorithm performance using a generic framework. *International Journal of Prognostics and Health Management*, Sep 2010.

(Feldman *et al.*, 2010b) Alexander Feldman, Gregory Provan, and Arjan van Gemund. Approximate model-based diagnosis using greedy stochastic search. *J. Artif. Int. Res.*, 38:371–413, May 2010.

(Freund and Schapire, 1995) Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, London, UK, 1995. Springer-Verlag.

(Hoos and Sttzle, 2004) Holger Hoos and Thomas Sttzle. *Stochastic Local Search: Foundations & Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

(Huang and Darwiche, 2005) Jinbo Huang and Adnan Darwiche. On compiling system models for faster and more scalable diagnosis. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 1*, pages 300–306. AAAI Press, 2005.

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

(Niggemann *et al.*, 2009) Oliver Niggemann, Benno Stein, Thomas Spanuth, and Heinrich Balzer. Using models for dynamic system diagnosis: A case study in automotive engineering. In *MBEES*, pages 46–56, 2009.

(Reiter, 1987) R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–96, 1987.

(Second International Diagnostic Competition (DXC 10), 2010) Second International Diagnostic Competition (DXC 10). Website, 2010. http://sites.google.com/site/dxcompetition2010/.

(Struss and Dressier, 1989) Peter Struss and Oskar Dressier. "physical negation": integrating fault models into the general diagnostic engine. In *IJCAI'89: Proceedings of the 11th international joint conference on Artificial intelligence*, pages 1318–1323, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

(Torta and Torasso, 2004) Gianluca Torta and Pietro Torasso. The role of OBDDs in controlling the complexity of model based diagnosis. In *15th International Workshop on Principles of Diagnosis (DX04)*, pages 9–14, 2004.

(Tsoumakas and Vlahavas, 2007) G. Tsoumakas and I. Vlahavas. Random k-Labelsets: An Ensemble Method for Multilabel Classification. In *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*, pages 406–417, Warsaw, Poland, September 2007.

(Tsoumakas *et al.*, 2010) Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–685. 2010.

(Williams and Ragno, 2007) Brian C. Williams and Robert J. Ragno. Conflict-directed a* and its role in model-based embedded systems. *Discrete Appl. Math.*, 155(12):1562–1595, 2007.