An Expert CAD Flow for Incremental Functional Diagnosis of Complex Electronic Boards

Cristiana Bolchini, *Senior Member, IEEE,* Luca Cassano, *Member, IEEE,* Paolo Garza, Elisa Quintarelli and Fabio Salice, *Member, IEEE,*

I. Introduction and Motivations

Functional diagnosis is used in industry to identify the source of an observed misbehavior during the operational life of a device. There are different motivations behind this kind of investigations, among which we mention 1) the identification of the faulty component for possible repair in case of an expensive board, hosting several subsystems, and 2) the collection of information on components' failure rates, to be able to monitor the quality of suppliers' products. Moreover, this second kind of information can also be used to improve efficiency and accuracy of the diagnosis process.

For diagnosis, a set of tests is executed and their outcomes are collected (called *syndrome*); based on the syndrome the process tries to identify the faulty component. A survey of approaches using "intelligent techniques" has been presented in [1], classifying them in rule-based and model-based ones. Approaches belonging to the former class use a set of rules in the form "IF syndrome THEN faulty component", whereas approaches in the latter class refer to a model of the system under diagnosis putting into relations faulty components and syndromes. Rules and models can either be directly specified by the test/diagnosis engineers or can be acquired by means of knowledge extraction methods, using records of previously executed diagnoses. In the latter case, an extensive research has been performed to identify the most effective machine learning technique, spanning from Bayesian inference [2], to decision trees [3], from artificial neural networks to support vector machines [4], [5]. A comparative analysis to evaluate the effectiveness of these techniques in extracting information from test-data volumes is proposed in [6]. Indeed it is important to be able to extract the knowledge from the previous historical data; however, we here focus on the subsequent diagnostic process, and its efficiency, in terms of the amount of information (test results) needed to identify and isolate the candidate faulty component. More precisely, we argue that it is fundamental to minimize the number of executed tests, to reduce testing time and associated costs. Thus, we propose an *incremental automatic functional diagnosis* process, that starting from a model (extracted or provided by the engineers), prompts the user to execute one of the available tests and based on the outcome, either identifies the cause of the failure, or selects a new test to be executed to proceed towards the diagnosis. As soon as enough evidence (test outcomes) is collected the process ends and the diagnosis is provided. This incremental process is designed to work in a scenario where traditional functional diagnosis is performed using the complete syndrome, therefore it faces the same difficulties. Indeed, the proposal aims at providing a solution that achieves the same results a diagnosis based on a complete syndrome would achieve (considered as the "golden" result), but with a reduced number of executed tests.

Incremental functional diagnosis was first presented in [7] using an engine based on Bayesian belief networks (BBN), achieving interesting results in terms of the number of tests necessary to perform the diagnosis. However, the identification of an initial set of tests and of a stop condition introduced two complex aspects, to be empirically determined for each system under consideration. Furthermore, different policies can be adopted to select the next test to be executed [8]. Indeed, the incremental approach is promising, however the use of the BBN engine does not allow for a systematic and automatic application and requires a supervised use by the diagnosis engineer, who drives the diagnosis based on his/her experience.

In this paper we propose a novel *incremental functional diagnosis* approach exploiting an automatic engine based on *data mining* (DM) achieving the same accuracy as the conventional diagnosis that uses complete syndrome, indicated as 100% accuracy. The method autonomously performs the diagnosis of a faulty system by minimising the number of

This work is partially supported by the Cisco University Research Program Fund – Gift #2012-101762 (3696), an advised fund of Silicon Valley Community Foundation.

Cristiana Bolchini, Luca Cassano, Elisa Quintarelli, Fabio Salice are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano. e-mail: firstname.lastname@polimi.it

Paolo Garza is with the Dipartimento di Automatica e Informatica, Politecnico di Torino. e-mail: firstname.lastname@polito.it

tests to be executed; based on the mined knowledge, not only the diagnosis is performed, but also additional diagnosis-related feedback is provided, e.g., redundant tests, masked components. In this paper we start from a model of the system under consideration provided by the test/diagnosis engineers; however, the approach can be used also after a preliminary phase devoted to extracting such model from a log of past syndromes (using data mining or the approaches proposed in [3], [4]), as it will be shown. The major contributions of the proposed methodology and companion framework are:

- a rule-based engine driven by rules putting into relation test outcomes and faulty candidates, automatically extracted from the system model;
- a completely automated CAD flow that identifies the candidate faulty component(s), without user's intervention or the need for empirically defined metrics;
- with respect to previous solutions, the possibility to achieve a 100% accuracy, lower number of executed tests to complete the diagnosis, and the possibility to leverage on such accuracy to further reduce the number of executed tests;
- information of the relative probability of being the candidate if more than one component is identified as faulty;
- feedback on the system model, to be exploited by the test and diagnosis engineers to improve test coverage and/or isolation in the system.

The paper is organized as follows. The next section presents existing approaches tackling the same problem or a similar one, to highlight their limitations and the open issues the proposed methodology and flow aims at solving. Section III introduces the background and the notions at the basis of the incremental diagnosis approach discussed in Section IV. The CAD flow implementing the methodology is presented in Section V, while the subsequent section reports the experimental results of the analysis and comparison with other similar solutions. Conclusions and future work close the paper.

II. Related Work

A number of works related to the use of machine learning and artificial intelligence for both incremental [7] and non-incremental [3], [4], [5] diagnosis can be found in literature.

The approach presented in [3] relies on decision trees (DTs). In the training phase, a DT is built starting from a set of complete syndromes and the associated faulty components (available from previous testing activities). Syndromes are represented as nodes in the DT, while candidate faulty components as leaves. In the diagnosis phase, the DT is fed with the actual *complete* syndrome and then is traversed until a leaf is reached, identifying the associated faulty component.

In [4] three adaptive functional diagnosis techniques based on artificial neural networks (ANNs), support-vector machines (SVMs) and weighted majority voting (WMV) between the such two techniques are presented. The paper first discusses how to use ANNs and SVMs to identify faulty components. Then it presents how to combine the two techniques through WMV to increase the achieved diagnostic accuracy. For each technique a weight expressing the confidence of the method on the considered board is calculated and then the majority voting uses these weights to rank the answers received from the two techniques to maximize the diagnostic accuracy. Building on the good performance of the SVMs-based method, the same authors propose in [5] a solution based on SVMs for an *incremental learning* of the component-test relation for performing the functional diagnosis process. The body of work presented in [3], [4] and [5] addresses the problem of building/*learning* an accurate model of the relation between *complete* syndromes and faulty components leading to the syndrome with a limited number of available bindings between the two. The diagnostic accuracy achieved by these methods is always significantly below 100%, although it is a relevant result, since it is does not assume the existence of a model. In [9] a framework based on the Dempster-Shafer theory for ranking the possible causes of a failure either at the board- or at the circuit-level is presented. Finally, a number of techniques (e.g., those presented in [10], [11]) have been proposed for enhancing the accuracy and effectiveness of the extracted model by discarding redundant tests and suggesting the introduction of more selective tests.

A comparative analysis of artificial intelligence-based techniques for extracting the system model from test data volumes is presented in [6], evaluating the amount of data needed by the different methods with respect to the achieved accuracy of the final diagnosis, when considering *complete* syndromes.

In [7] a first proposal of an incremental diagnosis technique is presented. The technique builds a Bayesian belief network (BBN) representing the probabilistic relation between failing tests and faulty components, starting from a model provided by the test engineers. Given a *partial* syndrome the method determines the probability of each component to be the faulty one. Based on this information and the inherent information that the execution of a test may bring, the approach incrementally performs tests until the faulty component(s) can be considered as "identified". This technique suffers from the absence of a test execution order and of an explicit stop condition. Identifying the next test to be executed and determining when to stop the diagnostic process are tasks left to the diagnosis engineer's experience. In [8], [12], [13] the same authors address the specific issues to overcome these problems. However, the solutions to these open challenges are based on heuristics to be tuned and monitored by the expert diagnosis engineer, whose contribution is still fundamental, an aspect we want to overcome, by proposing a solution that requires no empirical metrics.

Altogether, in the overall diagnosis panorama, the attention has been devoted to build a solid model of the system starting from previously collected testing/diagnosis sessions ([3], [4], [5]), while the challenge of improving the diagnosis process in terms of reducing the efforts (tests and consequently time) has been only partially solved ([7]), as the approach requires a significant effort to drive the BNN engine.

The approach introduced in [14] is a first proposal to overcome such limitations; it considers an incremental engine using a set of rules extracted with data mining, and used in a straightforward manner. The solution achieved promising results, dealing with a simplified context of syndromes. We

build on such initial framework to present here a complete methodology and framework, characterized by the following original contributions:

1) an improved rule-based engine driven by rules mined from the system model and post-processed to obtain an efficient set;
2) a strategy for achieving a 100% accuracy;
3) the interaction with a BBN-based tool to a) reduce the number of executed tests, by leveraging accuracy, and b) compute the relative probability of being the faulty component when more candidates are identified.

III. Background

We here recall some notions at the basis of our proposal.

A. Data Mining

Data Mining is the process of analyzing large volumes of data for extracting not so self-evident and previously unknown information from it. Different algorithms have been proposed in the literature, e.g., itemset and association rule extraction [15], [16], classification [**?**], [17], and clustering [18]. In this paper we deal with association rules, that describe the frequent co-occurrence of sets of items in a large amount of collected data [15]. They have been initially exploited to identify correlations among items in the market basket data analysis context. However, they have been exploited also in other contexts (e.g., network traffic data analysis). The input data, in the association rule mining context, is a dataset D composed of a set of transactions of arbitrary length, where each transaction d in D is a set of items. In our context, a transaction d in D is the complete outcome of a diagnosis process, composed by the set of test outcomes and the associated faulty component(s). Given an arbitrary dataset D as input, the association rule mining problem consists in mining the set of rules of the form $X \Rightarrow Y$ that are frequent in D (i.e., it consists in mining the most frequent correlations among items in D). More formally, an association rule R is usually represented as an implication in the form $X \Rightarrow Y$, where X and Y are two arbitrary itemsets (i.e., sets of items), called *Antecedent* and *Consequent* of R respectively, such that $X \cap Y = \emptyset$ (i.e., X and Y are disjoint sets). The quality of an association rule is usually measured by means of *support* and *confidence*. The Support measure corresponds to the frequency of the set $X \cup Y$ in the dataset D (i.e., the percentage of transactions in D that contain both X and Y) while the confidence measure corresponds to the conditional probability of finding Y in D, having found X and is given by $sup(X \cup Y)/sup(X)$ (i.e., the confidence of the rule $X \Rightarrow Y$ is the percentage of transactions containing both X and Y among those transactions containing X). The association rule mining problem is a well-known problem in the data mining community and various efficient algorithms have been proposed to perform the mining, among which we mention FP-growth-like algorithms [16], and we refer to them.

In some contexts, each item is also associated with a weight. In this case, each transaction d in D is a set of pairs ($item_i$, $weight_i$), where $item_i$ is an item and $weight_i$ is its weight in d. The association rules mined from these type of datasets are called weighted association rules. These rules, similarly to the traditional ones, are in the form $X \Rightarrow Y$, where X and Y are two arbitrary set of items. However, a set of weighted measures that take into consideration also the weights associated with each item, are used to assess the quality of weighted rules. The weighted support of a rule is computed by combining the weights of the items in the transactions matched by the rule (i.e., the transactions containing both X and Y) and is given by

$$W.\ Sup(X \Rightarrow Y) = \sum_{d \in D | (X \cup Y) \subseteq d} f\_weight(X \cup Y, d)$$

where $f\_weight(I, d)$ is a function that computes the weight of the set of items I in d, by combining the weights in d of the items in I. Different functions can be used to compute the weight of a set of items I in a transaction d. Similarly to Cagliero et al [19], we used the minimum function

$$f\_weight(I, d) = min_{(item_i, weight_i) \in d | item_i \in I} weight_i.$$

Hence, the weight of an arbitrary set of items I in a transaction d is given by the minimum weight among those of the items I in d. The weighted confidence of a rule $X \Rightarrow Y$ is given by $W.\ Sup(X \cup Y)/W.\ Sup(X)$

In our scenario the "items" are failing tests and components and each transaction in the input dataset D is associated with one component C_i and contains the set of tests T_j than can fail if C_i is faulty and C_i itself. Each item related to a test T_j is associated with a weight representing the probability that T_j fails if C_i is faulty. Hence, we use *association rule* mining to infer correlations between failing tests and faulty components, thus the rules have the form $\{T_1 \dots T_h\} \Rightarrow C_i$, where $T_1 \dots T_h$ are tests and C_i is a component. The rule states that `IF` $\{T_1 \dots T_h\}$ fail `THEN` C_i is the faulty component. In Section IV we describe how such rules are mined from the system model and exploited to support functional diagnosis.

B. Functional Diagnosis

Through the life cycle of a device, from the design phase to the final operative product, there are two activities with the goal of correlating the desired behavior and the observed one: testing and diagnosis.

Testing designates an activity whose purpose is to verify that the behavior of the system/sub-system does not deviate from what is expected (deviations between "pre-defined golden results" and "actual results"). It is worth noting that testing does not aim at identifying the cause and/or location of fault leading to the deviation from the expected behavior; the goal is to detect the presence of a problem, that is *fault detection*.

Diagnosis, on the other hand, designates an activity whose goal is to identify the cause or location of the fault *because* a misbehaviour has been detected. Therefore, diagnosis aims at identifying the cause and/or the location of a fault by means of interpreting the information provided by the test applied to the system. The degree of accuracy is called "resolution of the diagnostic test" and it depends on the ability of the test to isolate a fault. In general, it can occur that a fault cannot

be isolated, given the available test set. The cause could be controllability/observability problems related to the integration of different independent components on a board, such that test sets designed to fully exercise a component are not effective when the component is reachable through a chain of other components, or unforeseen interactions become manifest.

The sets of tests used for fault detection and diagnosis can coincide although in the former case the aim is to be able to stimulate the system so that a generic mismatch in the expected outcomes is obtained, while in the latter the user is interested in gathering as much information on the misbehaviour to be able to trace back the source of the problem. More precisely, test pattern generation aimed at fault detection identifies the minimal set of tests that allow to discover all possible faults. The higher the number of faults a test covers, the better. When performing diagnosis, it is important to distinguish among faults, therefore ideally, there should be a unique relation between a fault and a syndrome. However, this would imply that, in the worst case, all tests should be executed to find the faulty component. Therefore, the diagnosis test suite is usually bigger than the one used for testing purposes only. Finally, when performing *functional* diagnosis, the test procedure works by considering components as black-boxes, with no knowledge about their internals and usually performed at a high abstraction level, thus not taking into account manufacturing defects at a low-abstraction level.

Within this context, two working hypotheses are adopted.

- **At least a test fails.** This assumption refers to the concept of diagnosis, where the board is being investigated *because* a misbehaviour has been detected during the operational life of the device.
- **Single component failure.** It allows one to adopt an incremental diagnostic approach, stopping as soon as one faulty component is identified, otherwise it would be necessary to execute the entire test suite to identify *all* faulty components. The diagnosis, yet, may point to *a set of possibly* faulty components, called faulty *candidates*, due to limited isolation. In such a situation, even the complete syndrome is compatible with more than one component being faulty and it is not possible to exactly determine which one is *the* faulty one.

IV. The Proposed Incremental Diagnosis Approach

The incremental approach we propose, dubbed *FIND* (*Functional INcremental Diagnosis*), starts from a system model defined by the test/diagnosis engineer and automatically extracts from it (by means of data mining) a set of rules putting into relation the failure of the tests and the components being faulty. The incremental process starts from a processed sorted set of such rules ($\mathbb{RS}$) and iteratively selects the top ranking rule and requests the execution of the tests involved in the rule. Each outcome is collected and used to update the *partial syndrome* $\mathbb{PS}$, the set of candidates that – based on such outcome – cannot be faulty (not faulty components set, $\mathbb{NFCS}$), and the set of rules $\mathbb{RS}$. During the execution of the tests involved in the rule, it may happen that a rule does not hold anymore, and thus it is discarded, moving to the next rule

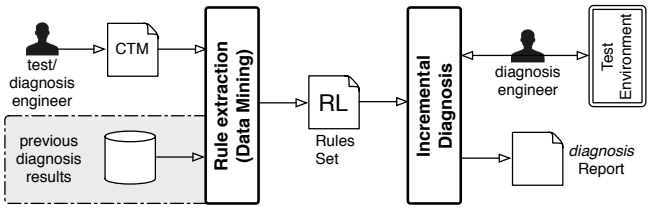

Fig. 1. The proposed incremental diagnosis approach.

(after all updates are performed). When all tests involved in the rule are performed without invalidating it (i.e., all tests in its antecedent fail), the rule is *satisfied* and it is exploited to determine the component (or sets of faulty components, $\mathbb{FCS}$) identified as faulty, concluding the process.

Figure 1 shows the proposed methodology, highlighting two alternative inputs for the extraction of the rules: the engineer's knowledge used to create a model or the log of past diagnoses. The extracted set of rules is strictly related to the Data Mining approach, as well as the strategies adopted in the incremental engine to process and exploit the rules according to their semantics. The pseudocode of this iterative algorithm based on the knowledge extracted with data mining is reported in Algorithm 1; all involved aspects are now discussed in detail.

```
1:  procedure FIND(CTM)
2:      FCS ← ∅
3:      NFCS ← ∅
4:      PS ← ∅
5:      RS ← GenerateRules(CTM)
6:      repeat
7:          R_i ← SelectTopRule(RS)
8:          TS_{R_i}    RetrieveTestSet(R_i)
9:          while R_i AND (TS_{R_i} ≠ ∅) do
10:             T_j    SelectTest(TS_{R_i})
11:             o_j    ExecTest(T_j)
12:             PS    UpdatePS(PS, o_j)
13:             NFCS ← updateNFCS(NFCS, o_j)
14:             RS    UpdateRS(RS, NFCS, o_j)
15:             TS_{R_i}    TS_{R_i} \ {T_j}
16:         end while
17:     until NOT satisfied(R_i)
18:     FCS ← RetrieveFaultyCandidates(R_i)
19:     return FCS, PS
20: end procedure
```

algorithm 1: FIND – Functional INcremental Diagnosis

A. System Model

For diagnosis purposes, a complex electronic board can be represented in terms of its components (e.g., microprocessors, DSPs, ...) and the set of tests that have been created to detect if faults have occurred. These tests are often defined for the specific component, in terms of sequences of input vectors and corresponding expected output responses. In general, for each component C_i a set of tests $\{T_h \dots T_k\}$ is defined to

verify the various functionalities implemented by C_i. When a test T_j is applied to the board inputs, the test *fails* when the values observed at the output are different from the expected ones; otherwise the test *passes*. Test engineers provide sets of tests such that if C_i is faulty, one or more of the tests in $\{T_h \dots T_k\}$ fail. However, when component C_i is integrated in a complex board, controllability/observability issues may rise, and furthermore, unforeseen interactions between components may occur, so that a test designed for a given component fails when a different component is faulty. As a consequence, test engineers can only provide a qualitative evaluation of the *probability that a test will fail when a component is faulty*. This model is similar to the one the diagnosis/test engineers define when using commercially available tools such as Agilent's Automatic Fault Detective tool (now discontinued) [20] and MonteJade [21]; thus the proposed approach can fit in a consolidated industrial diagnosis environment.

Based on these considerations, [7] introduces the *Components-Tests Matrix* (`CTM`) model we here adopt, that puts into relation components and tests. More precisely, rows represent components $\mathbb{C}$ and columns represent tests $\mathbb{T}$. Given a board having n_c components and n_t tests, `CTM` has $n_c \times n_t$ elements $ctm_{i,j}$, each one representing the probability that test T_j fails when component C_i is faulty. A discrete, qualitative scale is used for this information provided by test/diagnosis engineers. More precisely, they are required to specify how probable is that the test T_j fails when the component C_i is faulty, using a value in the set $\{\texttt{High}, \texttt{Medium}, \texttt{Low}, \texttt{None}\}$. Note that it is very difficult, or even impossible, to determine a quantitative estimation of this probability. Indeed, the reasons that may cause test T_j, designed for component C_i, not to fail when C_i is faulty or to fail when C_i is not faulty are related not only to component C_i itself but also to how the test is driven to the inputs of C_i and how test results are read back, e.g., if a JTAG interface is used to stimulate the component under test and to read back the results, and the JTAG interface itself is faulty, the outcome of the test execution will be `FAIL` even in case the component under test is not faulty. Furthermore, faults on the power supply or defective solderings on the board may interfere with the correct execution of the tests. It also worth noting that, as it has been demonstrated in [22], having a six-level set ($\{\texttt{High}, \texttt{MidHigh}, \texttt{Medium}, \texttt{MidLow}, \texttt{Low}, \texttt{None}\}$), instead of the previously presented four-level one, does not bring benefits to the analysis. The qualitative scale is then converted into a quantitative one:

$$ctm_{i,j} \in \{0.9, 0.5, 0.1, 0\}$$

Value 0.9 (0.1) instead of 1.0 (0.0) allows to maintain a degree of uncertainty due to components' interaction. This tolerance was designed for the BBN probabilistic reasoning engine of the original proposal. In our context it can be useful to model situations that rarely deviate from the expected behaviour (*outliers*, in the data mining scenario). On the other hand, should it be possible to define a model without such uncertainties, the performance of the proposed approach would even improve. Although arbitrary, the adopted quantitative scale has a limited impact on the methodology; a sensitivity analysis has been performed and reported in the experimental section, to show that the impact is limited to less than a 3% increase in the number of tests to be executed.

		Tests					
		T_1	T_2	T_3	T_4	T_5	T_6
Components	C_1	0.9	0.9	0.5	0.1	–	–
	C_2	0.5	–	0.9	0.1	0.5	–
	C_3	–	0.5	0.1	0.9	0.1	0.5
	C_4	0.1	0.1	–	–	0.9	0.9

Fig. 2. Sample Components-Tests Matrix used as a running example.

Throughout the paper we will refer to a running example `CTM`, shown in Fig. 2. Moreover, to exemplify the steps of the proposed approach we will behave as if the execution of the complete suite of tests would produce syndrome `PPPFFP`.

B. Association Rules Generation

The first step of our methodology is the use of data mining algorithms to extract association rules of the form $\{T_1 \dots T_h\} \Rightarrow C_i$ from a `CTM`. We will indicate the antecedent $\{T_1 \dots T_h\}$ of a rule R with $\mathcal{A}nt(R)$ and the consequent C_i with $\mathcal{C}ons(R)$; moreover we refer to the cardinality of $\mathcal{A}nt(R)$ as *length* of a rule (i. e., the number of involved tests). We mine a special type of association rules called *weighted association rules* [23], that also consider the importance of each item in a given dataset. The weights associated with the items of the input datasets are used to compute a *weighted support* measure that combines the frequency of a rule with the weights of its items. In particular, we use $ctm_{i,j}$ to assign an appropriate weight to each item, and to compute the weighted-measures (weighted support and confidence) according to the formulation presented in Section III-A. The input dataset D, from which the rules are mined, is composed of a number of transactions (i.e., rows) that is equal to the number of components. More specifically, for each component C_i there is one transaction $d_i \in D$ containing C_i and the set of tests T_j that should fail if C_i is faulty according to the `CTM` (i.e., the set of tests with a probability $ctm_{i,j} \neq 0$). Each test T_j in $d_j$is characterized by a weight equal to $ctm_{i,j}$.

As an example, consider tests T_1 and T_2. There are two rows where they appear at the same time, corresponding to C_1 and C_4 where they both have $ctm_{i,j} \neq 0$. The *unweighted* support of the pair $\{T_1, T_2\}$ is equal to 2. However, to take into consideration the probability values represented by $ctm_{i,j}$, we use them as weights: by summing 0.9 (the minimum between the probability value associated with T_1 and T_2 in the C_1 row) and 0.1 (the minimum in the C_4 row) we obtain a weighted support equal to 1.0. Moreover, the mined itemsets are combined to generate association rules and compute their confidence value. For instance, the confidence of the rule $\{T_1\ T_2\} \Rightarrow C_1$ is 90%, whereas the confidence of $\{T_1\ T_2\} \Rightarrow C_4$ is 10%, because the weighted support of $\{T_1, T_2\}$ is 0.9 in the C_1 row and 0.1 in the C_4 row.

The mining process, which infers the set of association rules $\mathbb{RS}init$, consists of two steps:

1) find all the sets of items (*itemsets*) whose weighted support exceeds a given threshold, and

2) generate the rules with a confidence greater than a given threshold, starting from the mined itemsets.

By enforcing thresholds for the minimum weighted support and the minimum confidence, the mining process is focused on the subset of rules that are "statistically" relevant, avoiding infrequent and not-representative rules. In our framework, the weighted association rule mining task is performed by means of a standard algorithm [19]; it mines all the association rules satisfying both thresholds, i.e., it is sound and complete with respect to the two enforced thresholds/constraints.

The set of association rules $\mathbb{RS}init$ is used, as described in the next section, to generate a rule set $\mathbb{RS}$ that includes *collapsed*, *sorted* rules that will be exploited by the incremental engine to identify the faulty component. Note that for each rule $\{T_1 \dots T_h\} \Rightarrow C_i$, we also compute the average and the variance of the weights in the CTM, between component C_i and tests T_j involved in the rule. More formally, the average of $\{T_1 \dots T_h\} \Rightarrow C_i$ is given by

$$Avg.Weight(\{T_1 \dots T_h\} \Rightarrow C_i) = \sum_{j=1}^{h} ctm_{i,j}/h$$

and the variance is computed on the same set of weights. These measures, together with rule confidence and length, are used during the rule ranking step.

76 association rules (some of which are shown in Fig. 3) are initially extracted from the running example CTM of Fig. 2.

As previously discussed, this approach differs from others in the same field (e.g., [3], [4]) because it focuses on the diagnostic process rather than on the identification of the relations between tests and components (i. e., the system model, here provided as an input). Indeed, we also investigated the possibility to extract the rules directly from the logs of previous diagnosis sessions, obtaining interesting results. In particular, we used data mining to directly extract the $\{T_1 \dots T_h\} \Rightarrow C_i$ rules, obtaining a set that is different from the one extracted from the CTM, but that leads to similar results in terms of accuracy and average number of executed tests. Thus, the approach can work even if a model is not provided and future work will further pursue this direction.

C. Rule collapsing and ranking

Once the association rules have been mined, a *collapsing* step is performed to group all rules with an Equivalent Antecedent, $\texttt{EA}(\cdot)$. The collapsed rules are built to populate a compact rule set $\mathbb{RS}$ by performing the following steps:

1) While $\mathbb{RS}init$ is not empty, starting from the first rule $R_i \in \mathbb{RS}init$, compute set

$$\texttt{EA}(R_i) = \{R_j \in \mathbb{RS}init | \mathcal{A}nt(R_j) = \mathcal{A}nt(R_i)\}$$

2) Build an *implication* (hereafter called *rule* without ambiguity, although it represents an aggregation of association rules) with the form:

$$R_{coll} = \mathcal{A}nt(R_i) \Rightarrow \bigcup \mathcal{C}ons(R_j)\ \forall R_j \in \texttt{EA}(R_i)$$

R_{coll} inherits the confidence, weighted support, average weight, and variance of rule $R_k \in \texttt{EA}(R_i)$ with the highest confidence.

#	Rules	Conf.	W. Supp.	Avg. Weight	Var.
1	$\{T_4\ T_6\} \Rightarrow C_3$	100.00%	50%	70.00%	400%
2	$\{T_1\ T_6\} \Rightarrow C_4$	100.00%	10%	50.00%	1600%
3	$\{T_3\ T_6\} \Rightarrow C_3$	100.00%	10%	30.00%	400%
4	$\{T_1\ T_2\ T_3\} \Rightarrow C_1$	100.00%	50%	76.66%	355%
		...			
9	$\{T_2\ T_4\ T_5\} \Rightarrow C_3$	100.00%	10%	50.00%	1066%
10	$\{T_1\ T_2\} \Rightarrow C_1$	90.00%	90%	90.00%	0%
11	$\{T_5\ T_6\} \Rightarrow C_4$	90.00%	90%	30.00%	400%
		...			
34	$\{T_4\} \Rightarrow C_3$	81.81%	90%	90%	0%
		...			
70	$\{T_1\ T_2\} \Rightarrow C_4$	10.00%	10%	10%	0%
71	$\{T_4\} \Rightarrow C_1$	9.09%	10%	10%	0%
72	$\{T_4\} \Rightarrow C_2$	9.09%	10%	10%	0%
		...			
76	$\{T_5\} \Rightarrow C_3$	6.67%	10%	10%	0%

Fig. 3. Portion of $\mathbb{RS}init$: association rules mined from the sample Components-Tests Matrix.

3) Compute $\mathbb{RS}init \setminus \texttt{EA}(R_i)$ then goto Step 1.

The rationale behind the definition of R_{coll} is that the best predictive rule is the one with the highest confidence, since confidence represents an estimate of the conditional probability that if all tests in the antecedent of the rule fail, the faulty component is the one in its consequent. Moreover, the collapsed rules are equivalent w.r.t. the set of tests to be performed to reach a prediction; indeed, when all the tests in R_k fail the rule is valid, thus our approach identifies as possible faulty components all the consequents of R_{coll}.

By referring to the running example in $\mathbb{RS}init$ (Fig. 3), association rules $\#34 : \{T_4\} \Rightarrow C_3$, $\#71 : \{T_4\} \Rightarrow C_1$ and $\#72 : \{T_4\} \Rightarrow C_2$ are collapsed into rule $\#34$ of Fig. 4 $\{T_4\} \Rightarrow C_1 \cup C_2 \cup C_3$. Hereafter, we use the ; symbol instead of $\cup$ in the rule consequent to compact the notation.

After the collapsing phase, a *ranking* algorithm is applied to identify the "best" predictive rules. In particular, an ordering based on confidence, rule length, support, average and variance is adopted. The best predictive rules are those with a high confidence, thus, we use that criterion first and if two rules have the same confidence value, the shortest one is preferred, to limit the number of test to be performed. The weighted support, the average and variance of weights are then subsequently used, in the reported order, to sort rules having the same values for the confidence and length measures. The shorter a rule is, the lower the number of tests to be executed is. However, a short rule is typically less discriminative/precise than a longer one, because the failure of a test is usually associated with many potential faulty components. For example, Rule $\#34$ of Fig. 4 has in its antecedent the unique item T_4. Based on this rule, if test T_4 fails there are three potential faulty components (i.e., C_1, C_2, and C_3). To discriminate among these components, at least one additional test is required (e.g., T_6). In fact, when considering test T_6, if it fails we can conclude that only C_3 is the faulty component, because Rule $\#1$ of Fig. 4, with confidence 100%, has only C_3 in its consequent. Thus, when ordering the mined rules, we first consider confidence and only when two rules have the same

#	Rules	Conf.	W. Supp.	Avg. Weight	Var.
1	$\{T_4\ T_6\} \Rightarrow C_3$	100.00%	50%	70.00%	400%
2	$\{T_1\ T_6\} \Rightarrow C_4$	100.00%	10%	50.00%	1600%
3	$\{T_3\ T_6\} \Rightarrow C_3$	100.00%	10%	30.00%	400%
4	$\{T_1\ T_2\ T_3\} \Rightarrow C_1$	100.00%	50%	76.66%	355%
5	$\{T_1\ T_3\ T_5\} \Rightarrow C_2$	100.00%	50%	63.33%	355%
6	$\{T_1\ T_2\ T_5\} \Rightarrow C_4$	100.00%	10%	36.66%	1422%
	...				
18	$\{T_1\ T_2\ T_3\ T_4\} \Rightarrow C_1$	100.00%	10%	60.00%	1100%
19	$\{T_1\ T_2\ T_5\ T_6\} \Rightarrow C_4$	100.00%	10%	50.00%	1600%
20	$\{T_1\ T_3\ T_4\ T_5\} \Rightarrow C_2$	100.00%	10%	50.00%	800%
	...				
34	$\{T_4\} \Rightarrow C_1; C_2; C_3$	81.81%	90%	10.00%	0%
35	$\{T_6\} \Rightarrow C_3; C_4$	64.28%	90%	50.00%	0%
	...				
41	$\{T_1\ T_4\} \Rightarrow C_1; C_2$	50.00%	10%	50.00%	1600%
42	$\{T_2\ T_5\} \Rightarrow C_3; C_4$	50.00%	10%	30.00%	400%
43	$\{T_4\ T_5\} \Rightarrow C_2; C_3$	50.00%	10%	30.00%	400%
	...				
48	$\{T_3\ T_4\} \Rightarrow C_1; C_2; C_3$	33.33%	10%	30.00%	400%

Fig. 4. $\mathbb{RS}$: collapsed and ranked rules generated from the $\mathbb{RS}init$ of Fig. 3.

confidence, we prefer the shortest one.

The outcome is a compact ranked rule set $\mathbb{RS}$ used by the engine to perform the incremental diagnosis process; the one for the adopted running example is partially shown in Fig. 4.

D. Rule and Test Selection

Given the list of collapsed and ranked rules $\mathbb{RS}$, FIND always selects the highest ranking rule R_{top} and the tests in its antecedent are executed, one at a time. Since the antecedent of an association rule is a conjunction of items, the *strength* of the rule is based on the "entire" set of involved tests, and it holds only if *all* tests fail. Therefore, it is not useful to impose an ordering in the execution of the tests involved in the rule. Once a test is executed, based on its outcome one of the following three alternatives occur: i) another test in the same rule needs to be executed, ii) the iterative process stops according to the conditions presented in Subsection IV-F, or iii) $\mathbb{RS}$ is updated as discussed in Subsection IV-E and the new top ranking rule is selected.

E. Not-Faulty Components Identification and Rules Pruning

After the outcome of a test has been collected, two activities can be carried out: i) rules pruning, and ii) not-faulty components identification.

Rules pruning is carried out after a test passes. This activity consists in invalidating all those rules involving the test that passes, because they will never hold.

$$\mathbb{RS} = \mathbb{RS} \setminus \{R_i | T_y \in \mathcal{A}nt(R_i) \wedge T_y = \texttt{PASS}\}$$

After every test execution, *not-faulty components identification* is carried out to populate the set of components that are not faulty ($\mathbb{NFCS}$, not-faulty component set). When the collected outcome is FAIL, all components having no relation ($ctm_{xy} = 0$) with such test can be considered as not faulty:

$$\mathbb{NFCS} = \mathbb{NFCS} \cup \{C_x | ctm_{xy} = 0 \wedge T_y = \texttt{FAIL}\}$$

When a test passes, following rule pruning, those components involved only in rules that have been pruned can then be considered as not faulty, and added to $\mathbb{NFCS}$.

Let us consider the first step of the iterative process, referring to the running example CTM, $\mathbb{RS}$ and selected PPPFFP syndrome. Rule #1 in Fig. 4 is the top ranking one, and test T_4 is executed, with outcome FAIL. Because $ctm_{4,4} = 0$, component C_4 cannot be the faulty component, thus $\mathbb{NFCS} = \{C_4\}$. Furthermore, C_4 is removed from all consequents and all rules with on C_4 in the consequent are pruned, causing an update of $\mathbb{RS}$. Then, test T_6 is executed, with outcome PASS. As a consequence, Rule #1 does not hold and all rules involving T_6 in the antecedent are pruned. The updated $\mathbb{RS}$, reported in Fig. 5, is then used and the new top ranking rule is selected to proceed with the process. The partial syndrome at this step is $\mathbb{PS} =$ ---F-P.

F. Stop Condition Evaluation

The process *should* stop as soon as i) enough information has been collected to take a decision, or ii) additional test outcomes do not add useful knowledge to take a more accurate decision. A main issue with the BBN approach presented in [7] is the ability of the system to automatically identify such a condition, letting the user decide empirically. In this work we have adopted two stop conditions: a *rule-based* one and a *BBN-based* one.

Rule-based stop condition: This is a systematic condition that leads to a diagnosis with a 100% accuracy and refers to the fact that each rule extracted by the data mining process intrinsically states that *IF the antecedent is satisfied THEN the consequent is true*, with a certain level of *confidence*. Therefore, since FIND always works on the top ranking rule, this stop condition guarantees that no additional knowledge is necessary to take a decision when a rule is satisfied. More precisely, an antecedent is satisfied when all tests FAIL, thus the iterative mechanism is interrupted when, given the top ranked rule

$$R_{top} : \{T_1 \dots T_h\} \Rightarrow C_x; C_y \quad (1)$$

all $T_1 \dots T_h$ fail and, as a consequence, C_x and C_y are considered as the faulty candidate components.

If there is a single component in the consequent, the process is concluded without further ado. If more components are involved in the consequent, the satisfied rule is of the form represented in Eq. 1.

Either C_x or C_y is the faulty component, but the executed tests involved in the rule ($T_1 \dots T_h$) do not allow to discriminate further.

With respect to the running example, the satisfied rule (with $\mathbb{PS}$ = PPPF-P) is

$$\{T_4\} \Rightarrow C_1; C_2; C_3$$

However, there could be longer rules (having more tests in the antecedent), with a lower confidence (thus not being the top rule), offering a more refined diagnosis that identifies only a subset of the candidate components involved in the satisfied rule. If the process stops here, we possibly erroneously identify a faulty component which is fault free (false positive), losing

#	Rules	Conf.	W. Supp.	Avg. Weight	Var.
1	$\{T_1\ T_2\ T_3\} \Rightarrow C_1$	100.00%	50%	76.66%	355%
2	$\{T_1\ T_3\ T_5\} \Rightarrow C_2$	100.00%	50%	63.33%	355%
3	$\{T_1\ T_4\ T_5\} \Rightarrow C_2$	100.00%	10%	36.66%	355%
4	$\{T_2\ T_3\ T_5\} \Rightarrow C_3$	100.00%	10%	23.33%	355%
5	$\{T_2\ T_4\ T_5\} \Rightarrow C_3$	100.00%	10%	50.00%	1066%
6	$\{T_1\ T_2\} \Rightarrow C_1$	90.00%	90%	90.00%	0%
7	$\{T_1\ T_5\} \Rightarrow C_2$	83.33%	50%	50.00%	0%
8	$\{T_2\ T_3\} \Rightarrow C_1; C_3$	83.33%	50%	70.00%	400%
9	$\{T_2\ T_4\} \Rightarrow C_1; C_3$	83.33%	50%	50.00%	1600%
10	$\{T_3\ T_5\} \Rightarrow C_2; C_3$	83.33%	50%	70.00%	400%
11	$\{T_4\} \Rightarrow C_1; C_2; C_3$	81.81%	90%	10.00%	0%
12	$\{T_1\} \Rightarrow C_1; C_2$	60.00%	90%	90.00%	0%
13	$\{T_2\} \Rightarrow C_1; C_3$	60.00%	90%	90.00%	0%
14	$\{T_3\} \Rightarrow C_1; C_2; C_3$	60.00%	90%	50.00%	0%
15	$\{T_5\} \Rightarrow C_2; C_3$	60.00%	90%	50.00%	0%
16	$\{T_4\ T_5\} \Rightarrow C_2; C_3$	50.00%	10%	30.00%	400%

Fig. 5. $\mathbb{RS}$ after executing $T_4 = \texttt{FAIL}$, $T_6 = \texttt{PASS}$; $\mathbb{NFCS} = \{C_4\}$, $\mathbb{PS}$ = `---F-P`.

accuracy. To achieve 100% diagnostic accuracy an additional mechanism, called *rule look-ahead* has been designed. This mechanism is triggered when the rule-based stop condition is satisfied and more than one component is involved in the consequent. It consists of the following steps:

1) Given the satisfied rule R_i, build the set of rules SR_i containing all the rules in $\mathbb{RS}$ involving a super-set of R_i antecedent and a sub-set of the R_i consequent.
2) If SR_i is empty, then stop.
3) If SR_i is not empty, then identify the top-rule R_{i_top} of SR_i.
4) Execute all the tests in R_{i_top} until either R_{i_top} is satisfied or a test passes.
   a) if R_{i_top} is satisfied, then go to step 1 with $R_i = R_{i\_top}$.
   b) if a test in R_{i_top} passes, then invalidate R_{i_top} and go to step 3

Indeed, it may occur that other rules exist (with a lower confidence) of the form

$$R_n : \{T_1 \dots T_h\ T_z\} \Rightarrow C_x \tag{2}$$

This rule highlights the presence of a tests (T_z) that may allow identifying C_x as the faulty component if its outcome is FAIL. Trying to satisfy such rules may refine the diagnosis at the cost of a higher number of executed tests.

With respect to the running example, the above discussed rule look-ahead mechanism leads to process rule

$$\{T_4\ T_5\} \Rightarrow C_2; C_3$$

Thus, by executing one additional test (T_5), the diagnosis is further refined.

BBN-based stop condition: This condition exploits information gathered from a Bayesian belief network, fed with the current partial syndrome and the set of not faulty components. More specifically, a BBN is built from the CTM model putting into relation components and tests, as proposed in [7] and shown in Fig. 6, where a network is created, with components being the *root causes* and tests the *evidence*, *observed* variables, and CTM entries are used to compute the conditional probabilities. Because the BNN is a complete model for the variables and their relationships, it can be used to *answer* probabilistic queries, such as the relative probability of each component to be the faulty one, with respect to the partial syndrome being observed (i.e., the subset of observed variables). Rather than using this information as the basis of the incremental engine (as in [7]), we here exploit it *only* to compute these probabilities. More precisely, we feed the BBN Evaluator with the CTM, $\mathbb{NFCS}$ and the $\mathbb{PS}$ and retrieve the probabilities. From the performed experiment, we observed that at the beginning of the diagnosis process, when few outcomes are known, the network is not *stable* and the sum of all these probabilities is higher than 1. However, as the information increases, the BBN computes probabilities such that the sum equals 1. Additional outcomes will further refine the diagnosis, and the computed probabilities may also change significantly, but the sum will remain 1. Thus, we defined the BBN-based stop condition to be satisfied as soon as the sum of the probabilities of all components not in $\mathbb{NFCS}$ equals 1.

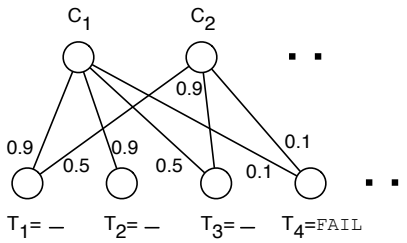

Fig. 6. BBN (portion of) built from the Components-Tests Matrix of Fig. 2.

Empirically, the amount of information gathered when the sum of all probabilities is 1 suffices to determine a diagnosis. Indeed, it may happen that components with a probability of being faulty greater than 0 could be ruled out by running further tests (false positives). This mechanism can be enabled if the user prefers to limit the number of tests, *betting* on an anticipated interrupt of the incremental process, with a *reasonable* confidence not to point out to too many false positive faulty components. Being it an empirically defined stop condition, the estimation of benefits and costs can be computed in general for a board, and may vary significantly. It is up to the user to decide whether to use it or not. We report in Sec. VI the effects of this additional stop condition.

G. Faulty Candidate Components Identification

When the process is completed, the identification of the faulty candidate set is performed by selecting all components involved in the satisfied rule, following the execution of the look-ahead mechanism, if the ruled-based stop condition is adopted. Indeed, if more than one component is identified, it is possible to query the BBN Evaluator with the CTM, the $\mathbb{NFCS}$ and the $\mathbb{PS}$ to retrieve a relative probability of each candidate. If the BBN-based stop condition is adopted, the faulty candidate set includes all components having a probability of being the faulty one higher than 0.

For the running example, the process stops according to the rule-based condition when $\mathbb{PS}$ = PPPFFP and $\mathbb{FCS} = \{C_2; C_3\}$. The query to the BBN Evaluator returns the additional information $\mathbb{FCS} = \{C_2[1.23\%]; C_3[98.77\%]\}$.

V. The Proposed Expert CAD Flow

The proposed methodology is supported by a CAD flow, presented in Fig. 7, consisting of various tools, namely the *SyndromeMaker* tool, the FIND tool and the BBN-Evaluator all implemented in C/C++. The flow is organised into two sub-flows: the *analysis sub-flow* and the *diagnosis sub-flow*.

A. The Analysis Sub-Flow

The analysis sub-flow is intended to be executed just once for a given board, to gather information on the diagnostic capability of the designed suite of tests and to prepare the sorted list of rules $\mathbb{RS}$ used by the incremental engine in the diagnosis sub-flow. More in detail, the analysis sub-flow consists of the execution of the *SyndromeMaker* tool and of the FIND tool working in a special *batch mode*. The SyndromeMaker, starting from the CTM, generates the list of all the legal syndromes, without any interaction with the user. For each legal syndrome the tool computes the probability of occurrence p_s, and the faulty component(s) associated with it. This information allows one to evaluate the average number of components that can be considered as faulty for every legal syndrome and identifies the components that cannot be isolated and the tests that are not useful for diagnostic purposes, i.e., tests that are never executed. If the process is performed at design time, it is possible to exploit the output of the analysis, i.e., the *system analysis report*, to improve tests' effectiveness and/or observability to increase the diagnostic resolution, as well isolation. As for the list of rules, this batch run can be used to remove from the initial $\mathbb{RS}$ all those rules that are never used, to work on a more compact $\overline{\mathbb{RS}}$ list.

By referring to the running example (CTM in Fig. 2), rule

$$\{T_1\ T_2\ T_6\} \Rightarrow C_4, 100.00\%, 10\%, 36.33\%, 1422\%$$

mined in the $\mathbb{RS}init$ is never used because it ranks below rule

$$\{T_1\ T_6\} \Rightarrow C_4, 100.00\%, 10\%, 50.00\%, 1600\%$$

When either T_1 or T_6 passes, both rules are pruned; when both T_1 or T_6 fail, the shorter rule is satisfied and thus the diagnostic procedure stops. Since there is a single component in the consequent (C_4) no rule look-ahead would be triggered. Thus the rule is removed.

In this perspective, mining rules and processing them to obtain $\overline{\mathbb{RS}}$ is an off-line activity, performed once; therefore the computational cost has a limited impact on the overall methodology. Finally, recall that if no CTM model is defined and only previous diagnosis outcomes are available, activities in the shaded area of Fig. 7 are replaced by the mining of rules directly from that information, to produce $\mathbb{RS}$.

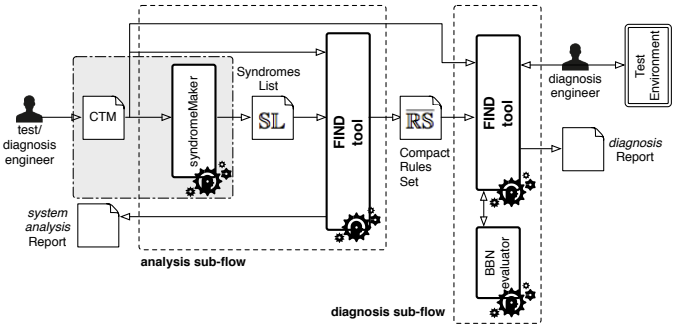

Fig. 7. The proposed CAD flow based on FIND.

B. The Diagnosis Sub-Flow

The diagnosis sub-flow is intended to be used by the engineers to perform diagnosis when a faulty board is brought back from service. During the execution of the diagnostic sub-flow, the FIND tool is used in *interactive mode*; at each step, the tool selects the test to be executed and waits for the engineer to provide the outcome. If the tool is integrated in an *Automatic* Test Environment, test request and outcome collection are performed automatically.

The process is iterated until a diagnosis is obtained, and, if the procedure leads to more than a single faulty component, a BBN Evaluator is invoked to get, based on the partial syndrome and the set of not-faulty components, the relative probability of each candidate to be the faulty component.

VI. Experimental validation

This section introduces the evaluation approach for the proposed methodology, followed by the setup for the experimental campaigns, reporting an in-depth discussion of the results.

A. Methodology evaluation

The relevance of the proposed methodology depends on two main aspects: i) the number of executed tests ($\#T$), and ii) the accuracy of the diagnosis ($Acc.$).

In fact, the goal is to reduce the effort for identifying the faulty component, possibly without compromising the accuracy. While the first aspect is intuitive, it is worth clarifying how accuracy is evaluated. Let us consider the complete syndrome selected as a running example: PPPFFP. According to the CTM of Fig. 2 the syndrome can occur when either C_2 or C_3 fail. The diagnostic outcome based on the complete syndrome would point to $\{C_2, C_3\}$ (we define them as components *associated with* that syndrome). The incremental approach we propose cannot further discriminate, either it computes the same diagnosis (possibly without executing all tests) or is inaccurate by finding a) a subset of the two associated components, or b) additional ones. Thus, the accuracy of the method is evaluated with respect to what a *traditional diagnosis based on complete syndromes* would find. It is possible to extract the outcomes of diagnosis based on complete syndromes by generating from the CTM all and only the *legal* syndromes, i.e., those combinations of test outcomes corresponding to a single component being faulty. For instance, in our running

example syndrome FFFFFF is not legal, since there is no single component that when faulty causes all tests to fail according to the CTM of Fig. 2.

Finally, given the CTM model, not all syndromes are equally likely to occur. For instance, according to our running example, when C_1 is faulty, syndrome FFFPPP is more likely to occur than PPPFPP, based on the estimated probabilities. Indeed, syndrome PPPFPP can also be caused by a failure in component C_2 or in C_3. Therefore, the total probability of occurrence of the syndrome is altogether not so low. Indeed, the lower the probability of a syndrome, the more difficult its diagnosis (especially if manually performed based on experience) and the more important the accuracy of the approach; we here anticipate that the proposed method always achieves a 100% accuracy, possibly with longer test sequences, although always well below the complete syndrome.

To evaluate the performance of the proposed methodology, we adopted the two mentioned indices $\#T$ and $Acc.$, weighted on the relative probability of occurrence of the syndrome; the impact of the behaviour of the approach for syndromes that very seldom may occur is less relevant than that of very frequent syndromes. We calculated these two indices as:

$$Acc = \sum_{s=1}^{n_{LS}} (acc_s \times p_s) \quad (3)$$

$$\#T = \sum_{s=1}^{n_{LS}} (num\_tests_s \times p_s) \quad (4)$$

where acc_s is the accuracy achieved by the method when analysing syndrome s, computed as follows:

$$acc_s = \frac{N_{Corr_s}}{N_{Tot_s} + N_{NotCorr_s}} \quad (5)$$

where N_{Corr_s} is the number of components correctly diagnosed, N_{Tot_s} is the total number of components associated with syndrome s and $N_{NotCorr_s}$ is the number of components wrongly diagnosed (either false positive or false negative); num_tests_s is the number of tests employed by the method when analysing syndrome s; finally, p_s is the probability of occurrence of syndrome s, computed as follows:

$$p_s = \sum_{i=1}^{n_c} \left(AFP_i \cdot \prod_{j=1}^{n_t} p_{i,j} \right) \quad (6)$$

where AFP_i is the a-priori failure probability of component C_i (note that for the sake of simplicity, but without loss of generality, we are assuming that components have all the same a-priori failure probability) and $p_{i,j}$ is defined as:

$$p_{i,j} = \begin{cases} 1 - ctm_{i,j} & if \quad s[T_j] = \texttt{PASS} \\ ctm_{i,j} & if \quad s[T_j] = \texttt{FAIL} \end{cases}$$

Where $s[T_j]$ represents the outcome of test T_j in syndrome s. In other words, $p_{i,j}$ represents the probability of test T_j having outcome $s[T_j]$ when the faulty component is C_i. Recall that it may occur that a given syndrome points to multiple faulty candidates that cannot be isolated. In such a situation, accuracy is 100% when the incremental approach identifies all and only the faulty candidates associated with such a syndrome, just as an approach using the complete syndrome would do.

TABLE I
BOARDS CHARACTERISTICS.

ID	n_C	n_T	n_{LS}	n_{FS}	n_{SS}	n_{RS}
1	4	9	98	17	36	45
2	5	13	415	19	66	330
3	7	17	2383	26	132	2225
4	10	14	1090	27	143	920
5	10	18	1678	56	94	1528
6	15	25	1249	19	109	1121
7	19	29	1339	22	134	1183
8	22	32	953	25	183	745
9	25	40	3063	18	141	2904
10	32	55	4677	15	196	4466
11	51	78	3292	16	189	3087
12	64	110	9354	2	212	9140

B. Experimental Setup

The accuracy and effectiveness of the proposed methodology and the efficiency of the CAD flow have been evaluated on a set of 12 synthetic (although realized according to realistic scenarios) boards of various sizes and complexity, whose characteristics are reported in Table I, in terms of number of components (column n_C), number of tests (column n_T) and number of legal syndromes (column n_{LS}) compatible with the CTM defining the board model. Legal syndromes have been clustered according to their occurrence probability to gather a more detailed idea of how the tool behaves when common and very unusual situations arise. The distribution of the syndromes is reported in the second part of Table I, where each entry indicates the number of *frequent* (column n_{FS}), *sporadic* (column n_{SS}) and *rare* (column n_{RS}) syndromes, having a probability of occurrence higher than 1%, between 1% and 0.1% and lower than 0.1%, respectively. The number of components ranges from 4 to 64, the number of tests from 9 to 110, and the list of legal syndromes (compatible with the CTM) has a cardinality varying from 98 to 9354.

C. Experimental Results and Discussion

The initial off-line application of the FIND flow allows to gather information on the board under test. We retrieve information (Table II) on the average number of faulty components associated with a syndrome ($avgFC$), the number of components that are *not isolated* from others (niC) and the number of tests that are *never used* during the diagnosis (nuT).

All experiments have been run by considering a complete syndrome at a time and knowing which one is the faulty component(s) causing such syndrome. Given such "hidden" syndrome, we used FIND to determine a diagnosis, by following the requested sequence of tests to be executed until the tool deems the diagnosis is reached.

As an example from CTM in Fig. 2, we consider syndrome FPPPFF, which has a probability of occurrence equal to $p_s = 1.8225\%$. When running the FIND tool for diagnosis, starting from rule #1 $\{T_4\ T_6\} \Rightarrow \{C3\}$ we execute test T_4 which, according to the hidden syndrome, passes ($\mathbb{PS}$ =

TABLE II
ANALYSIS SUB-FLOW RESULTS.

ID	avgFC	niC	nuT
1	1.49	1	0
2	1.72	2	0
3	1.84	3	0
4	2.22	0	0
5	2.46	0	0
6	2.29	5	0
7	2.44	6	0
8	2.11	3	0
9	2.34	12	0
10	2.03	4	0
11	2.41	15	0
12	2.03	8	0

TABLE III
EXPERIMENTAL RESULTS FOR THE PROPOSED AND EXISTING APPROACHES.

ID	DM (basic) [14]			BBN [7]			Proposed Approach		
	Acc.	#T	T[%]	Acc.	#T	T[%]	Acc.	#T	T[%]
1	100.00%	4.96	55.11%	75.04%	3.34	37.11%	100.00%	4.96	55.11%
2	100.00%	6.61	50.85%	83.62%	3.25	25.00%	100.00%	6.61	50.85%
3	99.99%	8.05	47.36%	93.29%	6.98	41.06%	100.00%	8.07	47.47%
4	99.96%	10.58	75.58%	90.10%	9.35	66.79%	100.00%	10.66	76.14%
5	99.72%	6.89	38.28%	76.19%	7.42	41.22%	100.00%	6.90	38.33%
6	98.67%	12.93	51.72%	95.69%	16.22	64.88%	100.00%	13.24	52.96%
7	99.09%	15.39	53.07%	94.79%	21.64	74.62%	100.00%	15.65	53.97%
8	99.72%	15.93	49.78%	95.22%	25.08	78.38%	100.00%	16.00	50.00%
9	99.87%	18.84	47.10%	93.67%	31.06	77.65%	100.00%	18.94	47.35%
10	99.63%	23.96	43.56%	92.98%	43.26	78.65%	100.00%	24.09	43.80%
11	99.33%	34.75	44.55%	97.69%	75.49	96.78%	100.00%	34.92	44.77%
12	96.62%	43.09	39.17%	80.45%	104.96	95.42%	100.00%	43.22	39.29%
avg	99.38%	-	49.68%	89.06%	-	64.80%	100.00%	-	50.00%

$---\texttt{P}--$). Based on this outcome, $\mathbb{RS}$ is updated and rule $\{T_1\ T_6\} \Rightarrow \{C3\}$ is then considered, requesting the execution of test T_1, that fails ($\mathbb{PS}$ = $\texttt{F}--\texttt{P}--$). The rule still holds and test T_6 is executed, failing. The rule is satisfied, the look-ahead mechanism does not find any other matching rule and the process stops, with the following outcome: $\mathbb{PS} = \texttt{F}--\texttt{P}-\texttt{F}$, $\mathbb{FC} = C_4$, number of executed tests $\#T = 3$.

The process has been repeated for all legal syndromes, computing accuracy and number of executed tests. Table III reports these values and the ratio $T(\%)$ between $\#T$ and the total number of tests for the proposed approach, as well as, for the same incremental approach using reasoning engines based on Data Mining (DM, [14]) and Bayesian belief networks (BBN, [7]), respectively. Note that we compare against the only existing alternative incremental approaches because other solutions (e.g., [3], [4]) focus on model extraction.

As a first comment, the proposed approach achieves 100% accuracy, outperforming the previous solutions. This is obtained thanks to the so-called *look-ahead* mechanism introduced to remove false positives that were observed when running the DM approach proposed in [14] with the entire set of legal syndromes, rather than only the most frequent ones. To achieve perfect accuracy, the length of the test sequence slightly increases w.r.t. DM (0.36% on average). When compared against the BBN solution, test sequence length is comparable, for small boards, but scales much better for larger boards (6.94% is the average reduction).

On average, the proposed approach requires the execution of half of the diagnosis tests clearly leading to a significant reduction of the whole diagnosis time. Nevertheless, the actual saving depends on the specific tests being avoided and therefore cannot be estimated. For example, the execution of an exhaustive main memory test may take from ten minutes to one hour, while a CPU stability test may take some hours up to a day. Moreover, the overall time required to carry out a test often depends more on the time for set-up/instrumentation of the test environment than on the actual test execution.

A second campaign has been executed to evaluate the benefits of the BBN-based stop condition, in reducing test sequence length, leveraging on accuracy. Results are reported in Table IV. The first part of the table shows test information, and more specifically the absolute average number of tests (column #T), the percentage of tests (column T[%]) and the test length reduction (column TLR[%]) with respect of the basic approach. Average test length reduction is 4.61%. The second part of the table reports accuracy information, for frequent, sporadic and rare syndromes. For frequent syndromes the approach behaves much better than for sporadic and rare ones; in particular, average accuracy values are 90.95%, 87.05% and 81.54%, respectively.

TABLE IV
BBN-BASED STOP CONDITION: TEST LENGTH IMPACT.

ID	Test information			Accuracy information		
	#T	T[%]	TLR[%]	Frequent	Sporadic	Rare
1	3.34	37.11%	18.11%	80.16%	67.99%	52.56%
2	4.94	38.00%	12.85%	92.04%	83.49%	70.63%
3	7.36	43.30%	4.06%	100.00%	98.19%	94.50%
4	10.13	72.38%	3.76%	96.89%	93.09%	90.56%
5	6.34	35.22%	3.11%	93.75%	84.85%	72.43%
6	12.68	50.72%	2.24%	99.12%	98.38%	91.30%
7	14.89	51.34%	2.63%	94.90%	93.53%	91.34%
8	15.66	48.94%	1.06%	96.28%	97.31%	91.47%
9	18.31	45.78%	1.57%	95.09%	93.17%	90.95%
10	23.26	42.29%	1.51%	93.52%	92.17%	88.40%
11	33.13	42.47%	2.30%	82.95%	73.12%	77.31%
12	40.85	37.13%	2.16%	66.67%	69.28%	66.99%
avg	-	45.39%	4.61%	90.95%	87.05%	81.54%

To better perceive the actual loss of accuracy, in Table V we report the number of syndromes (N) and their cumulative probability (CP) for which the Bayesian-based stop condition achieves a 100% diagnosis (0 false positives, 0FP), incurs in 1 false positive (1FP), and incurs in more than 1 false positives ($>$ 1FP). The average cumulative probability for the above values are 80.91%, 13.74% and 5.59%, respectively, which means that in 13.7% of cases, the diagnosis identifies an additional fault-free component, rarely more than one.

Although it goes beyond the scope of the work here presented, we also explored the behaviour of the proposed methodology and DM engine in the situation where the CTM model of the board under analysis is not provided, and only previous diagnosis logs are available. In this context, rules are directly mined from the logs, producing – for all boards – sets of rules with slightly different weights, usually leading to a sorted $\mathbb{RS}$ different from the one extracted from the CTM. We observed an average diagnostic accuracy of 99.72%, with

TABLE V
BBN-BASED STOP CONDITION: ACCURACY LOSS ANALYSIS.

ID	0FP		1FP		> 1FP	
	N	CP	N	CP	N	CP
1	22	56.05%	76	43.96%	0	0%
2	170	79.99%	245	23.01%	0	0%
3	2059	97.23%	324	2.77%	0	0%
4	842	89.29%	244	10.01%	4	0.61%
5	819	73.65%	859	26.35%	0	0%
6	805	93.15%	430	6.67%	14	0.18%
7	1187	87.33%	133	10.23%	19	2.44%
8	865	93.58%	80	4.14%	8	2.28%
9	2466	87.26%	577	5.36%	20	7.38%
10	3358	85.91%	1253	8.12%	66	5.97%
11	2308	68.34%	838	9.44%	146	22.22%
12	5236	59.13%	3769	14.83%	349	26.04%
avg	-	80.91%	-	13.74%	-	5.59%

TABLE VI
SENSITIVITY ANALYSIS TO CTM SCALE.

ID	$< 0.8, 0.2 >$		$< 0.7, 0.3 >$	
	#T	ΔT[%]	#T	ΔT[%]
1	5.24	3.11%	5.48	5.78%
2	6.73	0.92%	6.86	1.91%
3	8.32	1.47%	8.58	3.00%
4	10.55	0.78%	10.44	1.57%
5	7.97	5.95%	8.95	11.39%
6	13.14	0.40%	13.01	0.92%
7	15.54	0.39%	15.48	0.59%
8	15.92	0.25%	15.94	0.19%
9	19.03	0.23%	19.24	0.75%
10	23.64	0.82%	23.41	1.24%
11	34.89	0.04%	34.90	0.03%
12	42.31	0.83%	41.71	1.37%
avg	-	1.27%	-	2.40%

an average percentage of executed tests of 51.92% (the same values for the experiment based on the CTM are 100% average accuracy and 50.00% average number of executed tests). These outcomes confirm that the approach can be adopted also starting from existing logs of previous testing/diagnostic activities, and possible tuning is current ongoing work.

D. Robustness Analysis

Three campaigns have been carried out to analyse the sensitivity of the approach to 1) the adopted qualitative scale, 2) model inaccuracies related to erroneously estimated H, M and L values, and 3) model incompleteness related to missing component-test relations.

1) Sensitivity to the quantitative scale $\{0.9, 0.5, 0.1, 0\}$*.:* For each $\texttt{CTM}_\texttt{i}$ we defined two new CTMs $\texttt{CTM}_{\texttt{i}_{0.8}}$ and $\texttt{CTM}_{\texttt{i}_{0.7}}$, obtained from $\texttt{CTM}_\texttt{i}$ by changing the 0.9s into 0.8s and 0.7s, respectively and the 0.1s into 0.2s and 0.3s, respectively; 0.5s are left unaltered. We performed the same kind of experimental campaign with these new models (results are reported in Table VI). It can be observed that the adopted scale does not significantly affect the FIND behaviour. In particular, the accuracy is always 100% (therefore it is not reported), whereas the difference in terms of number of executed tests is very small in most cases: the average values are 1.27% for the $< 0.8, 0.2 >$ case and 2.40% for the $< 0.7, 0.3 >$ case. High differences can be observed only for the smallest CTMs characterised by several component-test dependencies.

2) Sensitivity to model inaccuracies.: We were interested in understanding how the method behaves when the engineer has erroneously estimated test-component relationships, having provided erroneous qualitative values (for instance a High instead of a Medium). To evaluate the impact of possible inaccuracies, we randomly corrupted a percentage of the entries; we defined two campaigns, one with a 10% of inaccurate values ($\texttt{CTM}_{\texttt{i}_{10\%}}$), the other with a 20% of inaccuracies ($\texttt{CTM}_{\texttt{i}_{20\%}}$). We only considered erroneous estimations (High and Low turned into Medium and viceversa), not contemplating the possibility of a missing/extra entry. The effect is a different order in $\mathbb{RS}init$ and different $\mathbb{RS}$ with respect to the one corresponding to the real board, which could lead to loss of accuracy and/or increased number of executed tests. We ran the previously presented experiments on $\texttt{CTM}_\texttt{i}$ but we evaluated the obtained results by calculating the Acc and $\#Tests$ indices with the syndrome occurrence probabilities (p_s) calculated from $\texttt{CTM}_{\texttt{i}_{10\%}}$ and $\texttt{CTM}_{\texttt{i}_{20\%}}$. Table VII shows the results: once more, accuracy is not affected (and therefore not reported) and the difference in the number of executed tests is very small: the average values are 1.21% and 2.39% for the 10% and the 20% inaccuracy cases, respectively, while the highest values are 4.54% and 6.11% in the two situations.

TABLE VII
ROBUSTNESS ANALYSIS AGAINST MODEL INACCURACIES.

ID	10% Inaccuracies			20% Inaccuracies		
	#T	T[%]	ΔT[%]	#T	T[%]	ΔT[%]
1	4.89	53.33%	0.78%	4.99	55.44%	0.33%
2	7.20	55.38%	4.54%	6.41	49.31%	1.54%
3	8.34	49.06%	1.59%	8.93	52.53%	5.06%
4	10.73	76.64%	0.50%	11.00	78.57%	2.43%
5	7.64	42.44%	4.11%	8.00	44.44%	6.11%
6	13.40	53.60%	0.64%	14.14	56.56%	3.60%
7	15.53	53.55%	0.41%	15.82	54.55%	0.59%
8	15.86	49.56%	0.44%	14.72	46.00%	4.00%
9	18.83	47.08%	0.28%	19.64	49.10%	1.75%
10	24.05	43.73%	0.07%	23.18	42.15%	1.65%
11	34.57	44.32%	0.45%	35.72	45.79%	1.06%
12	42.39	38.54%	0.75%	42.64	38.76%	0.53%
avg	-	50.60%	1.21%	-	51.10%	2.39%

3) Sensitivity to model errors.: Finally, we analysed how the approach behaves when the engineer omitted some component-test relationships. To evaluate this impact, we randomly removed a percentage of the entries; two campaigns have been executed, one removing 10% of the entries ($\texttt{CTM}_{\texttt{i}_{10\%}}$), the other removing 20% ($\texttt{CTM}_{\texttt{i}_{20\%}}$). The process takes as input a CTM with missing entries, leading to the extraction of $\mathbb{RS}init$ and $\mathbb{RS}$ differing from the "correct" ones. The modelled reality is the one where there are a number of existing syndromes that are not consistent with the CTM taken as input, and others for which a diagnosis can be performed, but potentially with inaccurate results. More specifically, all syndromes that cannot be explained by the erroneous CTM are identified, therefore – as soon as one occurs – the diagnosis engineer has a feedback on the erroneous model. Column 6 and 11 in Table VIII report the percentage of such *illegal* syndromes that allow such inconsistency identification, thus

TABLE IX
INDUSTRIAL CASE STUDIES CHARACTERISTICS.

ID	n_C	n_T	n_{LS}	n_{FS}	n_{SS}	n_{RS}
Board1	5	18	11788	14	72	11702
Board2	14	24	12290	16	20	12254
Board3	25	44	13261	17	39	13205

TABLE X
RESULTS FOR THE INDUSTRIAL CASE STUDIES.

ID	Acc	T[%]
Board1	100.00%	52.50%
Board2	100.00%	57.46%
Board3	100.00%	58.09%
avg	100.00%	56.01%

providing a suggestion on how fast it will be possible to detect the presence of the error (but not the specific error). We computed the accuracy with respect to the behaviour corresponding to having the correct CTM (columns 2 and 7), and with respect to a traditional approach using a complete syndrome *on the same erroneous model*; the accuracy is identical and therefore our solution achieves 100% accuracy with respect to the outcome using the complete syndrome, still requiring a reduced number of tests.

E. Industrial Case Studies

Finally, we applied the proposed methodology to three industrial boards (whose characteristics are summarized in Table IX in terms of number of components, tests, syndromes classified into frequent, sporadic and rare syndromes). One of this board is a micro-processor board, equipped with shared memory, arbiters, DMA controller, JTAG and other components. The other two are telecom medium-size boards whose CTMs have been designed at Cisco for systems diagnosis, using the Automatic Fault Detective tool [20].

We analysed all the legal syndromes of the three case studies. Results from this experiment are reported in Table X. It is worth noting that the proposed approach achieves a 100% diagnostic accuracy for all the considered industrial case studies. Furthermore, the number of tests needed to achieve a correct diagnosis is substantially reduced. More in detail, for all the considered boards the percentage of tests required to achieve the stop condition is about 50% (56.01% on average). The results collected on synthetic boards mimicking real ones to fully exercise the methodology are confirmed in the real-world industrial boards environment; the proposed approach successfully performs the diagnosis with a reduced number of executed tests, without compromising the diagnostic accuracy.

VII. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed an automatic incremental methodology and a CAD flow, based on data mining, to perform functional diagnosis of complex systems. The incremental approach allows to identify the faulty component with a 100% accuracy by performing a subset of all available tests. To further reduce the number of executed tests (and thus limiting costs) we also introduced an empirical condition based on Bayesian belief networks, leveraging on accuracy. An extensive experimental campaign has been performed, analysing the achieved results from different perspectives, to evaluate the efficiency of the proposed solution with respect to previous ones, showing significant improvements in terms of accuracy and of test length when leveraging on accuracy. The approach can also be used to gather information on the system being modelled with respect to diagnosis-related aspects (i. e., isolation) providing feedback to the test/diagnosis engineer.

Ongoing work is investigating the opportunity to mine the rules driving the incremental engine directly from existing diagnosis logs, either to support the engineers in providing the CTM model or as the starting point for the incremental functional diagnosis process. Furthermore, we plan to extend the analysis subflow to increase the amount of information provided to designers in the system analysis report.

REFERENCES

[1] W. G. Fenton, T. M. McGinnity, and L. P. Maguire, "Fault diagnosis of electronic systems using intelligent techniques: a review," IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 31, no. 3, pp. 269–281, Aug. 2001.

[2] Z. Zhang, Z. Wang, X. Gu, and K. Chakrabarty, "Board-level fault diagnosis using bayesian inference," in Proc. VLSI Test Symp., 2010, pp. 244–249.

[3] F. Ye, Z. Zhang, K. Chakrabarty, and X. Gu, "Adaptive board-level functional fault diagnosis using decision trees," in Proc. Asian Test Symp., 2012, pp. 202–207.

[4] ——, "Board-level functional fault diagnosis using artificial neural networks, support-vector machines, and weighted-majority voting," IEEE Trans. CAD Integrated Circuits and Systems, vol. 32, no. 5, pp. 723–736, 2013.

[5] ——, "Board-level functional fault diagnosis using multikernel support vector machines and incremental learning," IEEE Trans. CAD Integrated Circuits and Systems, vol. 33, no. 2, pp. 279–290, 2014.

[6] H. Wang, O. Poku, X. Yu, S. Liu, I. Komara, and R. D. Blanton, "Test-data volume optimization for diagnosis," in Proc. Design Automation Conf., 2012, pp. 567–572.

[7] L. Amati, C. Bolchini, L. Frigerio, F. Salice, B. Eklow, A. Suvatne, E. Brambilla, F. Franzoso, and M. Martin, "An incremental approach to functional diagnosis," in Proc. Int. Symp. Defect and Fault Tolerance in VLSI Systems, 2009, pp. 392–400.

[8] L. Amati, C. Bolchini, and F. Salice, "Test selection policies for faster incremental fault detection," in Proc. IEEE Intl Symp Defect and Fault Tolerance in VLSI Systems, Oct 2010, pp. 310–318.

[9] H. Fang, K. Chakrabarty, Z. Wang, and X. Gu, "Diagnosis of board-level functional failures under uncertainty using dempster-shafer theory," IEEE Tran. CAD Integrated Circuits and Systems, vol. 31, no. 10, pp. 1586–1599, Oct 2012.

[10] F. Ye, K. Chakrabarty, Z. Zhang, and X. Gu, "Information-theoretic framework for evaluating and guiding board-level functional-fault diagnosis," IEEE Design & Test, vol. 31, no. 3, pp. 65–75, 2014.

[11] Z. Sun, L. Jiang, Q. Xu, Z. Zhang, Z. Wang, and X. Gu, "Agentdiag: An agent-assisted diagnostic framework for board-level functional failures," in Proc. Int. Test Conference, Sept 2013, pp. 1–8.

[12] L. Amati, C. Bolchini, F. Salice, and F. Franzoso, "Improving fault diagnosis accuracy by automatic test set modification," in Proc. Int. Test Conf., November 2010.

[13] ——, "A formal condition to stop an incremental automatic functional diagnosis," in Proc. Euromicro Conf. Digital System Design, Architectures, Methods and Tools, 2010, pp. 637–643.

[14] C. Bolchini, E. Quintarelli, F. Salice, and P. Garza, "A data mining approach to incremental adaptive functional diagnosis," in Proc. Int. Symp. Defect and Fault Tolerance in VLSI and Nanotechnology Systems, 2013, pp. 13–18.

[15] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in Proc. Int. Conf. Very Large Data Bases, 1994, pp. 487–499.

TABLE VIII
ROBUSTNESS ANALYSIS AGAINST MODEL ERRORS.

ID	10% Errors					20% Errors				
	complete syndromes on correct CTM	complete syndromes on erroneous CTM				complete syndromes on correct CTM	complete syndromes on erroneous CTM			
	Acc.[%]	Acc.[%]	#T	T[%]	# Illegal Syn.	Acc.[%]	Acc.[%]	#T	T[%]	# Illegal Syn.
1	82.44%	100%	6.29	69.89%	12 (12.24%)	75.00%	100%	5.37	59.67%	32 (32.65%)
2	73.47%	100%	7.73	59.46%	120 (28.92%)	49.26%	100%	7.38	56.77%	201 (48.43%)
3	77.18%	100%	9.22	54.24%	1492 (62.61%)	50.62%	100%	9.24	54.24%	1189 (49.90%)
4	83.94%	100%	10.55	75.36%	318 (29.17%)	63.88%	100%	10.25	73.21%	445 (40.83%)
5	69.99%	100%	8.01	44.50%	564 (33.61%)	35.35%	100%	7.86	43.67%	990 (59.48%)
6	71.47%	100%	14.60	58.40%	428 (34.27%)	68.67%	100%	13.41	53.64%	649 (51.96%)
7	76.26%	100%	15.72	54.21%	283 (21.14%)	68.89%	100%	13.59	46.86%	688 (51.38%)
8	84.89%	100%	15.87	49.59%	168 (17.63%)	69.26%	100%	15.53	48.53%	433 (45.44%)
9	79.98%	100%	19.63	49.08%	934 (30.49%)	63.24%	100%	18.74	46.85%	1243 (40.58%)
10	79.31%	100%	23.69	43.07%	1719 (36.75%)	66.89%	100%	23.37	42.49%	3012 (64.40%)
11	78.16%	100%	33.50	42.95%	588 (17.86%)	64.05%	100%	33.00	42.31%	1644 (49.94%)
12	76.52%	100%	42.83	38.94%	2523 (26.97%)	55.69%	100%	41.59	37.81%	5168 (55.25%)
avg	77.80%	100%		53.31%		60.90%	100%		50.50%	

[16] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in Proc. Int. Conf. Management Data, 2000, pp. 1–12.

[17] E. Baralis, L. Cagliero, and P. Garza, "Enbay: A novel pattern-based bayesian classifier," IEEE Trans. Knowledge and Data Engineering, vol. 25, no. 12, pp. 2780–2795, 2013.

[18] J. Huang, H. Sun, Q. Song, H. Deng, and J. Han, "Revealing density-based clustering structure from the core-connected tree of a network," IEEE Trans. Knowledge and Data Engineering, vol. 25, no. 8, pp. 1876–1889, 2013.

[19] L. Cagliero and P. Garza, "Infrequent weighted itemset mining using frequent pattern growth," IEEE Trans. Knowl. Data Eng., vol. 26, no. 4, pp. 903–915, 2014.

[20] Fault Detective, 4.0, Agilent Technologies. [Online]. Available: www.agilent.com/find/fd

[21] L. Bardford, V. Kanevsky, and L. Kamas, "Bayesian fault diagnosis in large-scale measurement systems," in Proc. Instrumentation and Measurement Technology Conf., 2004, pp. 1234–1239.

[22] L. Amati, "Test and Diagnosis Strategies for Digital Devices: Methodologies and Tools," Ph.D. dissertation, Politecnico di Milano, 2012.

[23] W. Wang, J. Yang, and P. S. Yu, "Efficient mining of weighted association rules (WAR)," in Proc. Int. Conf. Knowledge discovery and data mining. ACM, 2000, pp. 270–274.