Unsupervised Root-Cause Analysis for Integrated Systems

Renjian Pan[1], Zhaobo Zhang[2], Xin Li[1], Krishnendu Chakrabarty[1] and Xinli Gu[2]
[1]Department of Electrical and Computer Engineering, Duke University, Durham, NC
{renjian.pan, xinli.ece, krishnendu.chakrabarty}@duke.edu
[2]Futurewei Technogies, Inc., Santa Clara, CA
{zzhang1, xgu}@futurewei.com

***Abstract*—The increasing complexity and high cost of integrated systems has placed immense pressure on root-cause analysis and diagnosis. In light of artificial intelligent and machine learning, a large amount of intelligent root-cause analysis methods have been proposed. However, most of them need historical test data with root-cause labels from repair history, which are often difficult and expensive to obtain. In this paper, we propose a two-stage unsupervised root-cause analysis method in which no repair history is needed. In the first stage, a decision-tree model is trained with system test information to roughly cluster the data. In the second stage, frequent-pattern mining is applied to extract frequent patterns in each decision-tree node to precisely cluster the data so that each cluster represents only a small number of root causes. In additional, L-method and cross validation are applied to automatically determine the hyper-parameters of our algorithm. Two industry case studies with system test data demonstrate that the proposed approach significantly outperforms the state-of-the-art unsupervised root-cause analysis method.**

I. Introduction

With the rapid development of communication technologies, network systems have become increasingly complex [1]. This brings more challenges to the design of diagnostic systems in order to identify the root causes of system failures rapidly and accurately. Once errors are detected, we can leverage monitoring techniques to collect various types of data (e.g., logging information, system metrics, etc.) [2]. However, it remains a challenge to pinpoint the root cause out of many candidates. The diagnosis process may take several days. Therefore, there is a pressing need to develop efficient and intelligent root-cause analysis methods.

During the past few decades, machine learning has been widely used in manufacturing and maintaining electronic systems [4]-[6]. In recent years, a number of smart diagnosis techniques based on machine learning have been proposed for integrated circuits, boards and systems [7]-[14]. Instead of fully relying on expert knowledge and manual diagnosis, these methods extract valuable information from data (e.g., manufacturing data, test data, and diagnostic data) and diagnose the root cause intelligently.

In general, there are two categories of root-cause analysis methods based on machine learning [7]: (i) supervised root-cause analysis, and (ii) unsupervised root-cause analysis. The supervised methods utilize supervised-learning models and utilize data of manually diagnostic history as training labels. In contrast, unsupervised methods utilize unsupervised-learning models and do not need any training label.

Most methods proposed in the literature focus on supervised root-cause analysis, because it naturally aligns with the common practice of training with labeled historical records and usually performs well in industrial diagnosis tasks. Popular supervised learning methods such as support-vector machines (SVMs), artificial neural networks (ANNs), and decision trees (DTs) have been successfully applied to board-level and system-level root-cause analysis [8]-[12].

Although supervised root-cause analysis has received much attention, it is not effective for all scenarios, especially when labels are insufficient or inaccurate. In modern integrated systems with complex architectures, it is not possible for experts to predefine all the failure classes. Moreover, new failure scenarios appear when different configurations are utilized, and systems are integrated in various ways. Thus, more dynamic and less human-dependent, unsupervised root-cause analysis methods have been proposed [7], [13].

These unsupervised methods automatically cluster the failure classes and mine the representative features in each class from the test/syndrome data without requiring diagnostic labels. An unsupervised feature learning method via ANNs has been proposed for mechanical systems with time-series data [13]. A clustering method based on self-organizing maps has been proposed for root-cause analysis of network systems [7]. However, the effectiveness of these methods is limited because the clustering is mainly implemented with features (i.e., test items). Without considering the correlation between test items and system defects, all test items are treated equally while clustering. Therefore, the important test items that are highly correlated with system defects cannot be efficiently identified. To address this problem, additional manufacturing/test information related to system defects is required.

In modern integrated systems, root-cause analysis can be performed at multiple levels. At the highest level, the pass/fail information (i.e., whether the system meets the specifications) is identified. If detailed test results are available at sub-system level, fine-grained root causes can be recommended. Although

such a fine-grained diagnosis procedure is time-consuming and label-intensive, the high-level pass/fail information is relatively easy and cheap to obtain. Taking a network communication system as an example, we can easily identify whether the network latency satisfies the service-level agreement. However, it is much more expensive to examine whether the failure is caused by bandwidth limitation, defects in the customer-premises equipment, or defects in the network server. In practice, the number of all possible root causes can be extremely large (e.g., more than one hundred [14]) before diagnosis, even though only a small portion of them may appear in defective systems after diagnosis. Therefore, it is prohibitively expensive to analyze each possible root cause by running the corresponding tests. If a machine-learning method can effectively utilize high-level pass/fail information and cluster failures into appropriate groups, the performance of unsupervised root-cause analysis will be greatly improved.

In light of the high-level pass/fail information, we propose an unsupervised root-cause analysis method for integrated systems. It is composed of two stages. In the first stage, a decision-tree model is built with system-level test data including pass/fail information to roughly cluster the data. In the second stage, frequent-pattern mining [15] is applied to extract frequent patterns in each decision-tree node to precisely cluster the data so that each cluster represents only a small number of root causes. In addition, *J*-fold cross validation [22] and L-method [23] are utilized to automatically choose the effective hyper-parameters for our proposed algorithm. Using two industry case studies, we show that the proposed approach is effective for root-cause analysis and significantly outperforms state-of-the-art unsupervised methods.

The remainder of this paper is organized as follows. In Section II, we present the problem formulation related to unsupervised root-cause analysis for integrated systems. In Section III, we propose a two-stage unsupervised learning method based on a decision-tree model and frequent-pattern mining to analyze the root cause of system failures. Experimental results on two industry case studies are presented in Section IV to validate the efficacy of the proposed method. Finally, we conclude the paper in Section V.

II. Problem Formulation

The overall flow of unsupervised root-cause analysis is summarized in Fig. 1. It is composed of two phases. In the training phase, unsupervised learning is applied to historical test data with high-level pass/fail information to learn a root-cause analysis model. Next, in the diagnosis phase, this model makes root-cause suggestion for a diagnosis system based on the test data.

We define $\boldsymbol{S}_L = \{S_{L,1}, S_{L,2}, \ldots, S_{L,N}\}$ to denote a set of samples of a system with historical test data $\boldsymbol{D}_L = \{\boldsymbol{d}_{L,1}, \boldsymbol{d}_{L,2}, \ldots, \boldsymbol{d}_{L,N}\}$ and pass/fail information $\boldsymbol{y}_{PF} = \{y_{PF,1}, y_{PF,2}, \ldots, y_{PF,N}\}$, where N is the number of samples for training. For each sample $S_{L,n}$, the vector $\boldsymbol{d}_{L,n} = \{d_{L,n,1}, d_{L,n,2}, \ldots, d_{L,n,T}\}$ denotes its test data with T test items. In this paper, we assume that the test data are numerical, instead of categorical. The parameter $y_{PF,n}$ is either 0 or 1, representing the pass/fail information that $S_{L,n}$ is defect-free or defective.

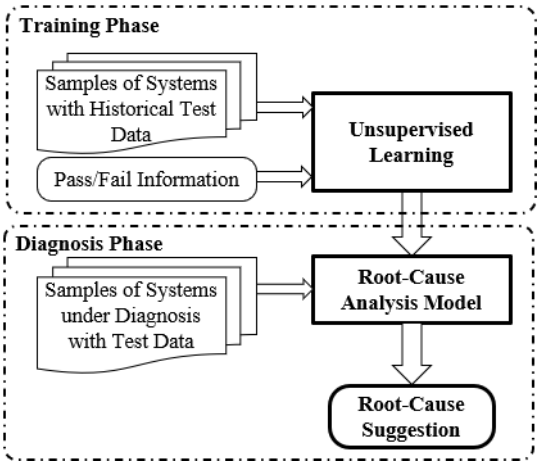

Fig. 1. The overall flow of unsupervised root-cause analysis.

In the diagnosis phase, we define $\boldsymbol{S} = \{S_1, S_2, \ldots, S_M\}$ to denote the set of samples of a diagnosis system with test data $\boldsymbol{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_M\}$, where M is the number of these samples. The root causes of these M defective samples form a set $\boldsymbol{r} = \{r_1, r_2, \ldots, r_M\}$. Our objective is to identify the correct root cause for each sample.

Based on the above definitions, the unsupervised root-cause analysis problem can be formulated as a clustering problem, where a clustering model is trained with historical test data $\boldsymbol{D}_L$ and pass/fail information $\boldsymbol{y}_{PF}$. Based on this clustering model and the test data $\boldsymbol{D}$, the samples in $\boldsymbol{S}$ of a diagnosis system are classified into different clusters as shown in Fig. 2, where each cluster represents only a small number of root causes. It is noteworthy that if we do not consider the pass/fail information $\boldsymbol{y}_{PF}$, the aforementioned problem formulation is equivalent to the traditional root-cause analysis in the literature [7].

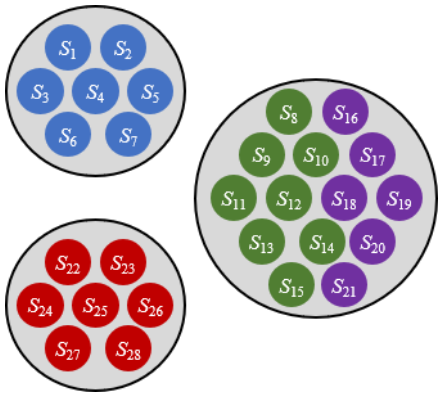

Fig. 2. A clustering example for the samples in $\boldsymbol{S}$ of a diagnosis system where four colors (i.e., blue, green, purple and red) represent four root causes, and three gray circles represent three clusters suggested by root-cause analysis.

We define the clustering results using the vector $\boldsymbol{c} = \{c_1, c_2, \ldots, c_M\}$, where c_n is the cluster label for the sample S_n. The clustering accuracy can be quantitatively assessed by the

normalized mutual information (NMI) between the clustering results $\boldsymbol{c}$ and the actual root causes $\boldsymbol{r}$ [16]:

$$\mathrm{I}_{NMI}\left(\boldsymbol{r},\boldsymbol{c}\right)=\frac{\mathrm{I}_{MI}\left(\boldsymbol{r},\boldsymbol{c}\right)}{\left(\mathrm{H}\left(\boldsymbol{r}\right)+\mathrm{H}\left(\boldsymbol{c}\right)\right)/2}, \tag{1}$$

where the numerator $\mathrm{I}_{MI}(\boldsymbol{r}, \boldsymbol{c})$ is the mutual information (MI) between $\boldsymbol{c}$ and $\boldsymbol{r}$, and the denominator is a normalized term defined by $\mathrm{H}(\boldsymbol{r})$ (i.e., the entropy of $\boldsymbol{r}$), and $\mathrm{H}(\boldsymbol{c})$ (i.e., the entropy of $\boldsymbol{c}$). The mutual information $\mathrm{I}_{MI}(\boldsymbol{r}, \boldsymbol{c})$ measures the common information (i.e., similarity) between the clustering results $\boldsymbol{c}$ and the actual root causes $\boldsymbol{r}$. Dividing $\mathrm{I}_{MI}(\boldsymbol{r}, \boldsymbol{c})$ by the average of $\mathrm{H}(\boldsymbol{r})$ and $\mathrm{H}(\boldsymbol{c})$ yields a normalized value over the internal [0, 1], indicating the "relative similarity" between $\boldsymbol{c}$ and $\boldsymbol{r}$. Hence, a large value of $\mathrm{I}_{NMI}(\boldsymbol{r}, \boldsymbol{c})$ implies accurate clustering results $\boldsymbol{c}$, given the actual root causes $\boldsymbol{r}$.

In this paper, we choose NMI as our performance measure because it not only considers the actual root causes $\boldsymbol{r}$, but is also able to accommodate multiple root causes in each cluster [16]. Other metrics proposed in the literature cannot appropriately assess the clustering performance for our application. For instance, the Davies-Bouldin score [17] does not take into account the actual root causes $\boldsymbol{r}$, thereby making the accuracy assessment less objective. The completeness score [18] assumes a single dominant root cause in each cluster and, therefore, is not applicable when a cluster may be associated with two or more root causes.

III. Proposed Method

Most conventional machine-learning methods are ill-equipped to attack our application of interest. On one hand, conventional supervised-learning methods such as support vector machine are not suitable because they need a large amount of historical test data labeled with expert-diagnosed root causes for training. On the other hand, conventional unsupervised-learning methods such as hierarchical clustering are not applicable either, as they do not take into account the pass/fail information that is critical for root-cause analysis. In this section, we develop a new method for accurate root-cause analysis to mitigate the above limitations of conventional approaches.

Our key idea is to combine conventional supervised and unsupervised learning methods to form a two-stage clustering framework, as shown in Fig. 3. In the first stage, supervised learning is adopted to build a decision-tree model to roughly cluster the data by carefully selecting the important test items that are highly correlated to the pass/fail information. However, the resulting decision tree cannot capture all the relevant test items for root-cause analysis because the pass/fail labels do not carry the full information of root causes. To address this issue, frequent-pattern mining is further applied in the second stage to precisely cluster the data and identify the other important test items ignored by the decision-tree model. After the Stage-II clustering is complete, each cluster is expected to correspond to only a small number of root causes.

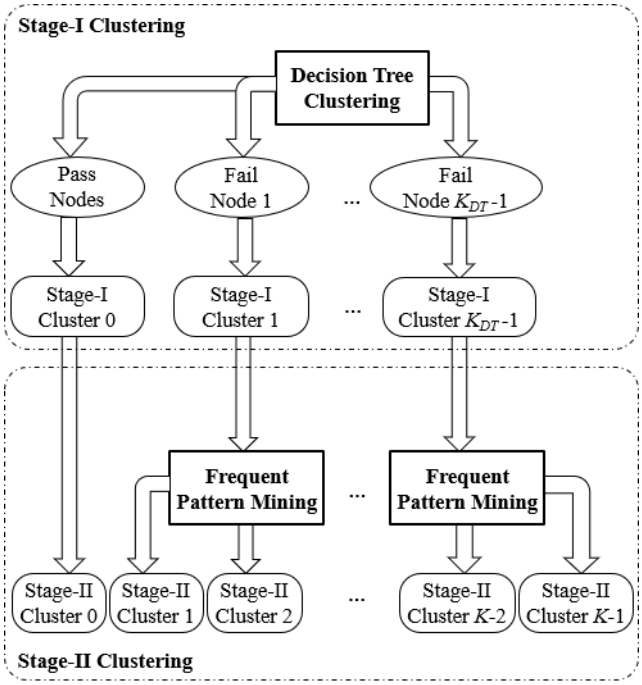

Fig. 3. Two-stage clustering for root-cause analysis.

A. Stage I: Rough Clustering Using Decision-Tree Learning

In the first stage of the proposed method, a decision-tree model is applied for rough clustering, as shown in the top box of Fig. 3. The decision-tree model has been shown as an effective and efficient method to utilize the pass/fail information and select the key features to diagnose whether a sample is defective [11]. Based on a set of historical test data labeled with pass/fail information, a tree-based structure is trained with multiple decision paths. Each decision path contains one or more decision rules. The historical test data are split into different leaf nodes with different decision paths. A decision tree can be treated as a clustering model because each leaf node naturally forms a cluster [20].

Based upon the above decision-tree model, the first stage of the proposed method is composed of two major tasks:

- *Decision-tree training*: A decision tree [22] is constructed by using the samples in $\boldsymbol{S}_L$ with the historical test data $\boldsymbol{D}_L$ and the vector of labels $\boldsymbol{y}_{PF}$. A leaf node is defined as the fail node, if most samples (i.e., over 50% of the samples) assigned to this node are defective. Otherwise, the leaf node is defined as a pass node. The hyper-parameters, such as the depth of the decision tree (T_{Dep}), can be automatically determined, as discussed in detail in Section III.C.
- *Stage-I clustering*: For the samples in $\boldsymbol{S}$ of a diagnosis system, the test data $\boldsymbol{D}$ are taken as the input of the decision tree. Following the decision path, every sample is assigned to one of the K_{DT} leaf nodes. As the samples in $\boldsymbol{S}$ are defective, very few of them may be assigned to the pass nodes due to the classification error posed by the decision tree. These samples assigned to the pass nodes form an outlier cluster that is labeled as "Stage-I Cluster 0" in Fig. 3, and the proposed method will not be able to diagnose

their root causes. For other samples in $\boldsymbol{S}$ assigned to fail nodes, coarse-grained root causes are collected from the decision path. Samples assigned to the k-th fail node form the k-th Stage-I cluster, and Stage-II clustering will be further triggered to accurately extract the failure patterns, as discussed shortly in Section III.B.

The clusters generated by the decision tree typically do not meet the diagnostic resolution requirement, because only the selected test items on decision paths are considered. To address this problem, we apply frequent-pattern mining to further analyze the other test items in the second stage.

B. Stage II: Precise Clustering by Frequent-Pattern Mining

In the second stage, we keep the outlier cluster (i.e., "Stage-I Cluster 0" in Fig. 3) and label it as "State-II Cluster 0" in Fig. 3. We apply frequent-pattern mining to further split the Stage-I clusters associated with the fail nodes. We first introduce the concept of frequent-pattern mining and then explain its application in our method.

Frequent-pattern mining has been developed in the literature to identify the most frequent patterns in a given dataset. It has been broadly used for correlation mining, classification and clustering [19]. To explain the algorithm underlining frequent-pattern mining, we first introduce some key terminology. Assume that we have a dataset $\boldsymbol{D}_F$ with N_F samples and T_F categorical features. For the continuous features in our application, discretization is needed as discussed later.

Definition 1: *An **item** is defined by the specific value of one data feature and is represented as $d_{F,*,t} = x$, where t is the feature index and x is its value.*

Definition 2: *A **pattern** is a set of items and is represented as $\{d_{F,*,t1} = x_1, d_{F,*,t2} = x_2, \ldots\}$.*

Definition 3: *A **frequent item/pattern** is an item/pattern appearing at least N_{Th} times in the dataset, where N_{Th} is a given frequency threshold.*

TABLE I
AN EXAMPLE DATASET FOR FREUQNET-PATTERN MINING

Sample Index	$d_{F,*,1}$	$d_{F,*,2}$	Frequent Items
1	A	C	[$d_{F,*,2}$ = C, $d_{F,*,1}$ = A]
2	B	C	[$d_{F,*,2}$ = C, $d_{F,*,1}$ = B]
3	B	C	[$d_{F,*,2}$ = C, $d_{F,*,1}$ = B]
4	A	C	[$d_{F,*,2}$ = C, $d_{F,*,1}$ = A]
5	A	D	[$d_{F,*,1}$ = A]

TABLE II
EXTRACTED FREQUENT PATTERNS OF THE FP-TREE IN FIG. 4

Frequent Patterns	Frequency
{$d_{F,*,2}$ = C}	4
{$d_{F,*,1}$ = A}	3
{$d_{F,*,1}$ = B}	2
{$d_{F,*,2}$ = C, $d_{F,*,1}$ = A}	2
{$d_{F,*,2}$ = C, $d_{F,*,1}$ = B}	2

For test applications, an item represents a test item with a specific test result, a pattern is a set of test items (including the corresponding test results), and a frequent item/pattern stands for an item/pattern appearing at least N_{Th} times in the historical test data. From this point of view, the threshold N_{Th} is the minimum number of defective samples required to represent a root cause. Based on these definitions, a frequent-pattern tree (or FP-tree in short) is built to mine the frequent patterns. We use an example to illustrate how to build the FP-tree. Table I shows a dataset with two features $d_{F,*,1}$ and $d_{F,*,2}$.

- *Initialization*: Assuming N_{Th} = 2, a list of frequent items is extracted and sorted by their frequencies. In Table I, there are three frequent items: '$d_{F,*,1}$ = A', '$d_{F,*,1}$ = B' and '$d_{F,*,2}$ = C', and their frequencies are 3, 2 and 4 respectively. The item '$d_{F,*,2}$ = D' only occurs once and is not considered as a frequent item. Hence, the complete list of all frequent items is: $\boldsymbol{f}_{Item}$ = [$d_{F,*,2}$ = C, $d_{F,*,1}$ = A, $d_{F,*,1}$ = B].
- *FP-tree learning*: The frequent items are used to grow the FP-tree as shown in Fig. 4. Starting from the first sample, a child node v_1, corresponding to the "high-frequency" item '$d_{F,*,2}$ = C', is added under the root as shown in Fig. 4(a), where N(v_1) denotes the frequency of the node v_1 at the current iteration step. Another child node v_2, corresponding to the "low-frequency" item '$d_{F,*,1}$ = A', is added under v_1, as shown in Fig. 4(b). If the sample carries more than two frequent items, all frequent items should be sequentially added to the FP-tree, as ordered by their frequencies. After the first sample is processed, a similar learning process is conducted for the second sample by increasing the frequency of v_1 by 1, as the frequent item '$d_{F,*,2}$ = C' also appears in the second sample. In addition, a new child node v_3, corresponding to '$d_{F,*,1}$ = B', is added under v_1 as shown in Fig. 4(c). After learning from all data samples, the final FP-tree is shown in Fig. 4(d). Note that the final FP-tree is independent of the order in which the data samples are processed. The steps used to construct the FP-tree is summarized in Algorithm 1.

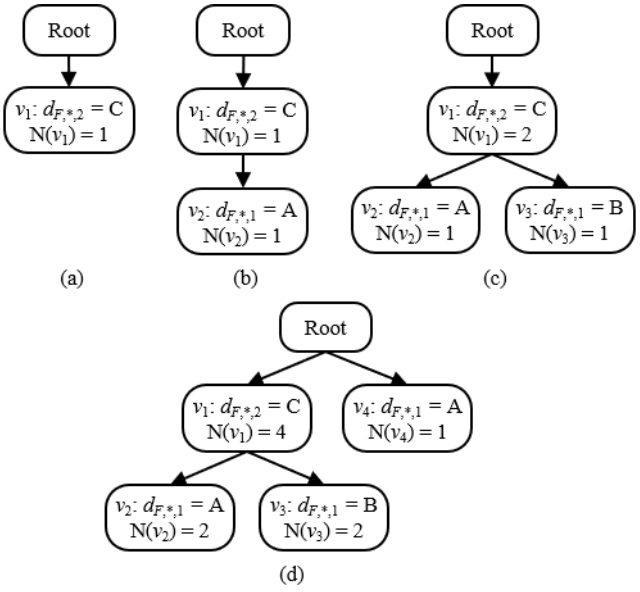

Fig. 4. An example of FP-trees: (a) after learning from the first frequent item $d_{F,*,2}$ = C of the first sample, (b) after learning from the first sample, (c) after learning from the first two samples, and (d) after learning from all samples.

Algorithm 1: FP-Tree Construction

1. Select the frequent items from the dataset $\boldsymbol{D}_F$ and sort them based on their frequencies as $\boldsymbol{f}_{Item} = [f_{Item,1}, f_{Item,2}, \ldots]$.
2. Initialize the FP-tree with a single root node v_{Root}.
3. For each data sample $\boldsymbol{d}_{F,n}$ in $\boldsymbol{D}_F$:
4. Select the frequent items in $\boldsymbol{d}_{F,n}$. Sort them according to the order of $\boldsymbol{f}_{Item}$ and form $\boldsymbol{f}_{Pattern,n}$.
5. Set the current FP-tree node: $v_{Current} = v_{Root}$.
6. For every $f_{Item,t}$ in $\boldsymbol{f}_{Pattern,n}$:
7. If $v_{Current}$ has a child node v_{Child} labeled as $f_{Item,t}$
8. Set $N(v_{Child}) = N(v_{Child}) + 1$.
9. Otherwise
10. Create a new child node v_{Child} labeled as $f_{Item,t}$, and set $N(v_{Child}) = 1$.
11. End If
12. Set $v_{Current} = v_{Child}$.
13. End For
14. End For

- *Pattern mining*: Based on the FP-tree, we extract the frequent patterns by the FP-growth algorithm to search the FP-tree in the depth-first manner and count the frequency for every tree node [19]. Taking the FP-tree in Fig. 4(d) as an example, we start from the root node, recursively search its children nodes (i.e., v_1 and v_4), and extract frequent patterns from the sub-trees. For the sub-tree beginning with the node v_1, the pattern $\{d_{F,*,2} = \text{C}\}$ is first identified and its frequency equals 4, as labeled by $N(v_1) = 4$ in Fig. 4(d). Next, by further visiting the two children nodes v_1 and v_4, four additional patterns $\{d_{F,*,2} = \text{C}, d_{F,*,1} = \text{A}\}$, $\{d_{F,*,2} = \text{C}, d_{F,*,1} = \text{B}\}$, $\{d_{F,*,1} = \text{A}\}$ and$\{d_{F,*,1} = \text{B}\}$ are found and their frequencies are all equal to 2. Finally, we visit the sub-tree beginning with the node v_4, and extract the pattern $\{d_{F,*,1} = \text{A}\}$ again. Hence, the frequency associated with $\{d_{F,*,1} = \text{A}\}$ is updated to 2 + 1 = 3. The complete list of all frequent patterns is shown in Table II. More implementation details for pattern mining can be found in [15].

It is noteworthy that the frequent items for each data sample are sorted in frequency-descending order in Algorithm 1. This ensures a higher likelihood for more prefix strings and structures being shared between data samples so that a simpler (i.e., with a less number of nodes) FP-tree can be built with less computational cost [15].

In this paper, we adopt the above frequent-pattern mining approach to identify the important test items that are ignored by the decision-tree model in Section III.A. If a test item is associated with a frequent pattern extracted from defective samples, it may contribute to the root-causes of interest. Based upon these additional test items identified by frequent-pattern mining, precise clustering is performed in the second stage of the proposed method, as summarized by the following steps.

- *Discretization*: We first discretize the entire historical test data $\boldsymbol{D}_L$ for $\boldsymbol{S}_L$ [19]. For each test item, the data are discretized into four bins by using the equi-depth strategy [21]. The test data $\boldsymbol{D}$ for the samples in $\boldsymbol{S}$ of a diagnosis system is also discretized with the same boundaries. In our future work, we will further explore other intelligent methods to optimally discretize the data.
- *Frequent-pattern mining*: As the samples in $\boldsymbol{S}_L$ with the historical test data $\boldsymbol{D}_L$ pass through the decision tree specified in Section III.A, we obtain the subsets $\boldsymbol{S}_{L,k}$ and $\boldsymbol{D}_{L,k}$ associated with the k-th Stage-I cluster ($k \geq 1$). To perform frequent-pattern mining on $\boldsymbol{D}_{L,k}$, only the test items ignored by the Stage-I decision tree are considered for mining because our goal is to extract valuable information from these unused test items for Stage-II clustering. While mining the frequent patterns and counting their frequencies against the given threshold N_{Th}, we only consider the defective samples in $\boldsymbol{S}_{L,k}$, because the root causes of interest should be learned from these samples. The threshold N_{Th} should be adaptively determined by the L-method that will be further discussed in Section III.C. To quantitively assess the importance of each frequency pattern, we calculate the failure rate for each frequent pattern $\boldsymbol{f}_{k,i}$ by taking into account both the defect-free and defective samples in $\boldsymbol{S}_{L,k}$:

$$\mathrm{P}_{Fail}\left(\boldsymbol{f}_{k,i}\right) = \frac{\mathrm{N}_{Fail}\left(\boldsymbol{f}_{k,i}\right)}{\mathrm{N}_{Fail}\left(\boldsymbol{f}_{k,i}\right) + \mathrm{N}_{Pass}\left(\boldsymbol{f}_{k,i}\right)}, \tag{2}$$

  where $\mathrm{N}_{Pass}(\boldsymbol{f}_{k,i})$ and $\mathrm{N}_{Fail}(\boldsymbol{f}_{k,i})$ represent the numbers of defect-free and defective samples in $\boldsymbol{S}_{L,k}$ sharing the frequent pattern $\boldsymbol{f}_{k,i}$, respectively. When a given frequent pattern $\boldsymbol{f}_{k,i}$ is shared by a large number of defective samples but only a small number of defect-free samples, the failure rate $\mathrm{P}_{Fail}(\boldsymbol{f}_{k,i})$ in (2) should be large. This denotes that the pattern $\boldsymbol{f}_{k,i}$ is high-correlated with defective samples, thereby implying its importance to our root-cause analysis. After mining, multiple frequent patterns $\boldsymbol{F}_k = \{\boldsymbol{f}_{k,1}, \boldsymbol{f}_{k,2}, \ldots\}$ are obtained for the k-th Stage-I cluster.
- *Frequent-pattern matching*: As the test data $\boldsymbol{D}$ for the samples in $\boldsymbol{S}$ pass through the decision tree specified in Section III.A, we obtain the subsets $\boldsymbol{S}_{I,k}$ and $\boldsymbol{D}_{I,k}$ associated with the k-th Stage-I cluster ($k \geq 1$). An effective frequent pattern $\boldsymbol{f}_{Eff,k,n}$ is found to match each sample $\boldsymbol{s}_{k,n} \in \boldsymbol{S}_{I,k}$. To do so, we check the failure rate in (2) for each pattern $\boldsymbol{f}_{k,i} \in \boldsymbol{F}_k$. The pattern $\boldsymbol{f}_{Eff,k,n}$ with the largest failure rate is selected among all frequent patterns that appear in sample $\boldsymbol{s}_{k,n}$.
- *Stage-II clustering*: Based on the matched frequent patterns $\{\boldsymbol{f}_{Eff,k,1}, \boldsymbol{f}_{Eff,k,2}, \ldots\}$, the samples in $\boldsymbol{S}_{I,k}$ with the test data $\boldsymbol{D}_{I,k}$ are further split into multiple Stage-II clusters, where samples with the same/different $\boldsymbol{f}_{Eff,k,n}$ are respectively assigned to the same/different Stage-II clusters.

After two-stage clustering, each Stage-II cluster is expected to include only a small number of root causes. For each Stage-II cluster, human experts may use the test information of the corresponding decision path of the decision tree and the corresponding frequent pattern to further investigate the underlying root causes.

C. Implementation Details

Traditionally, a decision tree is built with a given hyper-parameter T_{Dep} to specify its depth. Starting from the root node, we grow the decision tree by recursively splitting its nodes

based on the Gini impurity measure [22]. The decision tree stops growing when the given depth T_{Dep} is reached.

By implementing the traditional decision-tree algorithm, we may observe a number of fail nodes (i.e., Stage-I clusters) and the number of defective samples associated with each of these fail nodes is less than N_{Th}, thereby resulting in two critical issues. First, recall that N_{Th} denotes the frequency threshold for frequent-pattern mining and, hence, it defines the minimum number of defective samples required to represent a root cause. A cluster with less than N_{Th} defective samples is unrepresentative and should not be considered as a root cause. If a fail node $v_{DT,n}$ in the decision tree is associated with less than N_{Th} defective samples, no frequent pattern can be identified and, consequently, Stage-II clustering cannot be further performed at $v_{DT,n}$. Second, the decision tree may over-fit the training data $\boldsymbol{D}_L$, even if we appropriately set up the depth T_{Dep}. In other words, the decision tree may be split into too many leaf nodes corresponding to a large number of small clusters. In this case, it is desirable to develop a methodology to prevent the decision tree from creating too many leaf nodes.

To address the above problem, we modify the traditional decision-tree construction algorithm by taking into account the frequency threshold N_{Th}. When we consider the splitting of a node $v_{DT,n}$ into two child nodes $v_{DT,n,1}$ and $v_{DT,n,2}$ to grow the decision tree, we check whether $v_{DT,n,1}$ and $v_{DT,n,2}$ are fail nodes. If either $v_{DT,n,1}$ or $v_{DT,n,2}$ is a fail node and the number of defective samples associated with the fail node is less than N_{Th}, we do not split $v_{DT,n}$. In those cases, we can ensure that an overly small cluster will not be created after Stage-I clustering by the decision tree.

Based on these discussions, our proposed two-stage clustering method relies on two important hyper-parameters: (i) the depth T_{Dep}, and (ii) the threshold N_{Th}. To find the effective values of these two hyper-parameters, we first set $N_{Th} = 0$ and apply J-fold cross validation (CV) [22] to determine the effective depth T_{Dep}. In other words, we do not restrict the minimum number of defective samples in each fail node when we determine T_{Dep}. We treat the decision tree as a classifier to predict the pass/fail information of a given sample. Let $\{T_{Dep,1}, T_{Dep,2}, \ldots\}$ denote a set of possible values of T_{Dep}. We split the historical test data $\boldsymbol{D}_L$ into J folds $\{\boldsymbol{D}_{L,CV,1}, \boldsymbol{D}_{L,CV,2}, \ldots, \boldsymbol{D}_{L,CV,J}\}$. We also define $\mathrm{P}_{Acc}(j, T_{Dep,d})$ as the prediction accuracy on $\boldsymbol{D}_{L,CV,j}$ where the decision tree is trained by the other $(J-1)$-fold data with the depth $T_{Dep,d}$. The classification accuracy $\mathrm{P}_{Acc}(T_{Dep,d})$ is calculated as:

$$\mathrm{P}_{Acc}\left(T_{Dep,d}\right) = \frac{1}{J}\sum_{j=1}^{J}\mathrm{P}_{Acc}\left(j, T_{Dep,d}\right). \tag{3}$$

Once the classification accuracy $\mathrm{P}_{Acc}(T_{Dep,d})$ is expressed as a function of $T_{Dep,d}$, the effective depth T_{Dep} is chosen to maximize the accuracy $\mathrm{P}_{Acc}(T_{Dep,d})$.

Next, to determine the effective threshold N_{Th}, we note that N_{Th} directly influences the number of Stage-I clusters (i.e., K_{DT}) and the number of frequent patterns $\boldsymbol{F}_k = \{\boldsymbol{f}_{k,1}, \boldsymbol{f}_{k,2}, \ldots\}$ for the k-th Stage-I cluster. Hence, the total number of Stage-II clusters (i.e., K) is a function of N_{Th}. If a smaller threshold N_{Th} is chosen, more frequent patterns will be identified, and more stages-II clusters will be formed. Once the value of K is adaptively determined, the effective value of N_{Th} is uniquely found.

Motivated by this observation, we adopt the L-method [23] to determine the effective value of N_{Th} by optimizing K. Let $\{N_{Th,1}, N_{Th,2}, \ldots\}$ denote a set of possible threshold values of N_{Th}. For each value $N_{Th,d}$, we first train the decision tree with the given $N_{Th,d}$ and the effective depth T_{Dep} in the first stage. Next, we extract the frequent patterns with the given threshold $N_{Th,d}$ in the second stage. Once the Stage-II clusters are formed, we calculate the number of Stage-II clusters (i.e., K_d) and the intra-cluster distance $\mathrm{I}_{Dis,d}$ [24]:

$$\mathrm{I}_{Dis,d} = \sum_{k=1}^{K_d}\sum_{n=1}^{N_{d,k}}\left\|\boldsymbol{d}_{L,Fail,d,k,n} - \bar{\boldsymbol{d}}_{L,Fail,d,k}\right\|_2^2, \tag{4}$$

where $\|\bullet\|_2$ represents the L2-norm of a vector, $N_{d,k}$ is the number of samples in the k-th cluster, $\boldsymbol{d}_{L,Fail,d,k,n}$ denotes the test data for the n-th sample in the k-th cluster, and $\bar{\boldsymbol{d}}_{L,Fail,d,k}$ is the center of the k-th cluster:

$$\bar{\boldsymbol{d}}_{L,Fail,d,k} = \frac{1}{N_{d,k}}\sum_{n=1}^{N_{d,k}}\boldsymbol{d}_{L,Fail,d,k,n}. \tag{5}$$

Hence, we obtain a set of triplets $\{(N_{Th,1}, K_1, \mathrm{I}_{Dis,1}), (N_{Th,2}, K_2, \mathrm{I}_{Dis,2}), \ldots\}$.

Next, we plot the distance $\mathrm{I}_{Dis,d}$ as a function of K_d, as shown by the example in Fig. 5. A "knee point" can be found to determine the effective value of K by approximating the nonlinear function $\mathrm{I}_{Dis}(K_d)$ by a piecewise linear function consisting of two linear segments. To do so, we check each possible intersection point K_{Int} among $\{K_1, K_2, \ldots\}$, fit two linear segments by least-squares regression [22], and calculate the mean-squared error $\mathrm{I}_{MSE}(K_{Int})$ as a function of K_{Int}. The effective value of K is chosen to minimize the mean-squared error $\mathrm{I}_{MSE}(K_{Int})$.

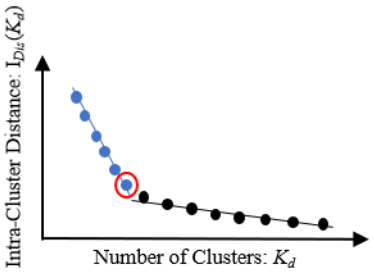

Fig. 5. An example of determining the effective number of clusters (i.e., K) by the L-method.

Taking Fig. 5 as an example, a blue line and a black line are both used to approximate $\mathrm{I}_{Dis}(K_d)$. The intersection point of these two lines, highlighted by a red cycle in Fig. 5, is the "knee point" corresponding to the effective number of clusters: $K = K_d$. In this example, if two different linear segments are used to approximate $\mathrm{I}_{Dis}(K_d)$, the intersection point is formed at a different location and the approximation error is expected to increase. Finally, given the effective value $K = K_d$, we

determine the effective threshold $N_{Th} = N_{Th,d}$ based on the corresponding triplet ($N_{Th,d}$, K_d, $I_{Dis,d}$).

D. Summary

Algorithm 2 summarizes the overall framework of the proposed unsupervised root-cause analysis method. It is composed of two phases: (i) the training phase, and (ii) the diagnosis phase. The training phase is based on the historical test data $\boldsymbol{D}_L$ for $\boldsymbol{S}_L$ as shown in Steps 1-5. The diagnosis phase processes the test data $\boldsymbol{D}$ for the samples in $\boldsymbol{S}$ of a diagnosis system as shown in Steps 6-8. The efficacy of our proposed approach will be demonstrated by two industry case studies in Section IV.

Algorithm 2: Unsupervised Root-Cause Analysis
Training phase:
1. Begin with the historical test data $\boldsymbol{D}_L$ for $\boldsymbol{S}_L$.
2. Determine the effective depth T_{Dep} by J-fold cross validation.
3. Determine the effective threshold N_{Th} by the L-method.
4. Train the decision tree for Stage-I clustering with the effective depth T_{Dep} and the effective threshold N_{Th}.
5. Extract the frequent patterns for the samples in $\boldsymbol{S}_{L,i}$ corresponding to each Stage-I cluster and form the Stage-II clusters with the effective threshold N_{Th}.

Diagnosis phase:
6. Begin with the test data $\boldsymbol{D}$ for the samples in $\boldsymbol{S}$ of a diagnosis system.
7. Form the Stage-I clusters by the decision tree.
8. Form the Stage-II clusters for the samples in $\boldsymbol{S}_{I,n}$ in each Stage-I cluster where each Stage-II cluster represents only a small number of root causes.

It is noteworthy that the proposed method is explainable, because both decision tree and frequent-pattern mining are explainable machine-learning models. For example, a decision path may be represented as $\{d_{*,1} < x_1, d_{*,2} \geq x_2\}$ by the decision tree, where $d_{*,1}$ and $d_{*,2}$ denote the test results of two test items, and x_1 and x_2 stand for the boundaries of these two test items. For frequent-pattern mining, a frequent pattern may be expressed as $\{\tilde{d}_{*,3} = \text{A}, \tilde{d}_{*,4} = \text{B}\}$ where $\tilde{d}_{*,3}$ and $\tilde{d}_{*,4}$ denote the test results of two test items after discretization, and A and B represent the discrete values of these two test items. The aforementioned information helps human experts understand how our machine-learning models make their decisions and, consequently, provide useful design insights.

IV. Experimental Results

In this section, we demonstrate the effectiveness of our proposed root-cause analysis using two case studies from industry. Both cases adopt real-world test data from network systems. Human experts have manually labeled the root causes of all defective samples by analyzing available test data and performing additional tests and/or repairments. The root causes labeled by human experts are considered as the ground truth to validate the performance of the proposed method. All experiments are performed on a computer with 2.6 GHz CPU and 16 GB memory.

For comparison purposes, we have further implemented a state-of-the-art method based on Ward's hierarchical clustering [7]. In our implementation, we enumerate the number of clusters (i.e., K_{HC}) from 1 to 100, considering the fact that the number of possible root causes should be much less than 100. Next, we select the optimal K_{HC} to maximize the normalized mutual information I_{NMI} in (1). Note that such an "optimal" K_{HC} cannot be reached in practice, because we do not know the "true" root causes. In other words, we attempt to compare the proposed method against the "best-yet-ideal" results that can be accomplished by the conventional method.

A. Network Product #1

In this example, 19 network system tests are applied to one network product with 2609 samples. These tests check the connectivity information (e.g., latency, package loss, etc.) of each sample. Four possible root causes are considered by human experts for this product. It is important to note that the actual number of root causes is unknown in practice and human experts cannot simply diagnose the root causes by running additional tests for each possible root cause. In this example, the aforementioned four root causes are used only for comparison purposes, and no root-cause information is needed by the proposed method. In our experiments, 1597 samples are labeled with the pass/fail information and they form the set $\boldsymbol{S}_L$ with the historical test data $\boldsymbol{D}_L$ for training purpose. No root-cause information is available for these 1597 samples. The other 2609 – 1597 = 1012 samples are defective. These samples form the set $\boldsymbol{S}$ of the diagnosis system, and their test data are denoted as $\boldsymbol{D}$ in the diagnosis phase.

For the proposed method, we apply 5-fold cross validation to select the effective depth T_{Dep} of decision tree among 19 possible depth values $\{1, 2, \ldots, 19\}$, given the fact that there are 19 test items in total. Fig. 6 shows the classification accuracy $P_{Acc}(T_{Dep,d})$ estimated by 5-fold cross validation as a function of the depth of the decision tree $T_{Dep,d}$. As highlighted by the red cycle in Fig. 6, the effective depth is $T_{Dep} = 4$, where the classification accuracy reaches the maximum value $P_{Acc}(T_{Dep}) = 0.90$.

We also apply the L-method to select the effective threshold N_{Th} among 24 possible values {1%, 2%, 3%, …, 10%, 12.5%, 15%, 17.5%, 20%, 22.5%, 25%, 30%, 40%, …, 100%}×1012, considering the fact that there are 1012 defective samples in total during the training phase. The lower bound is set as 1%, because any cluster with less than 1% ×1012 ≈ 10 defective samples should be unrepresentative for root-cause analysis. Fig. 7 shows the intra-cluster distance $I_{Dis}(K_d)$ as a function of the number of clusters K_d. Based on the L-method, the effective number of clusters is $K = 8$, as highlighted by the red cycle in Fig. 7. The corresponding effective threshold is N_{Th} = 8% ×1012 ≈ 81.

For the conventional hierarchical clustering method [7], Fig. 8 shows the normalized mutual information I_{NMI} as a function of the number of clusters K_{HC}. In our experiment, we choose the optimal number of clusters to maximize I_{NMI}. As highlighted by the red cycle in Fig. 8, such a "best-yet-ideal" result is achieved when K_{HC} equals 50.

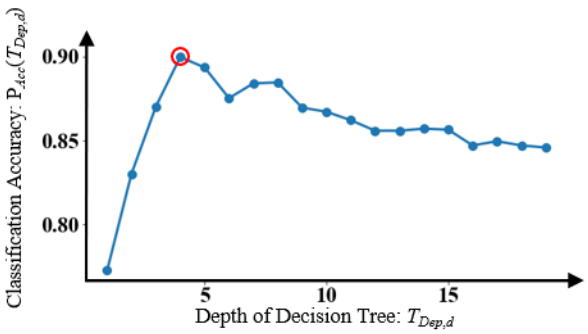

Fig. 6. Classification accuracy estimated by 5-fold cross validation as a function of the depth of the decision tree for Network Product #1, where the effective depth is highlighted by the red cycle.

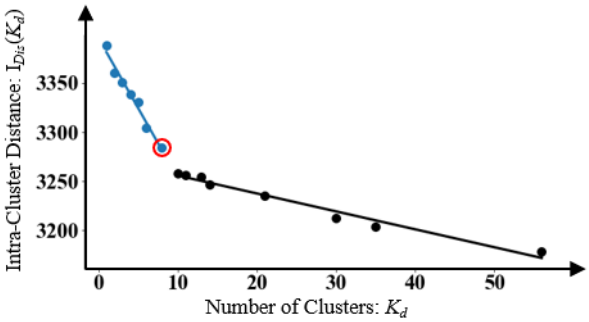

Fig. 7. Intra-cluster distance as a function of the number of clusters for Network Product #1, where the effective number of clusters, highlighted by the red cycle, is determined by the L-method.

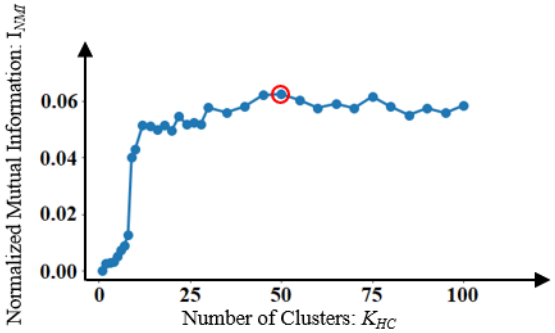

Fig. 8. Normalized mutual information as a function of the number of clusters for the conventional hierarchical clustering method [7] on Network Product #1, where the "best-yet-ideal" result is achieved with the optimal number of clusters as highlighted by the red cycle.

Based on the effective hyper-parameters: T_{Dep} = 4, N_{Th} = 81, K = 8 and K_{HC} = 50, both the conventional hierarchical clustering method [7] and the proposed approach are applied to the set ***S*** of the diagnosis system. After the decision tree is trained by our method, 58 defective samples (out of 1012 defective samples) in ***S*** are incorrectly assigned to the pass nodes, and they form the outlier cluster (i.e., "Stage-I Cluster 0" in Fig. 3).

Table III compares the conventional hierarchical clustering method [7] with the proposed approach for root-cause analysis. The normalized mutual information I_{NMI} in (1) is used as the performance of interest for comparison. As shown in Table III, the proposed method achieves substantially higher I_{NMI} than the conventional approach, while the runtime for both methods is less than one minute.

TABLE III
PERFORMANCE OF ROOT-CAUSE ANALYSIS ON NETWORK PRODUCT #1

Method	I_{NMI}	K	Runtime (s)
Hierarchical clustering [7]	0.062	50	10.7
Proposed method	0.293	8	19.0

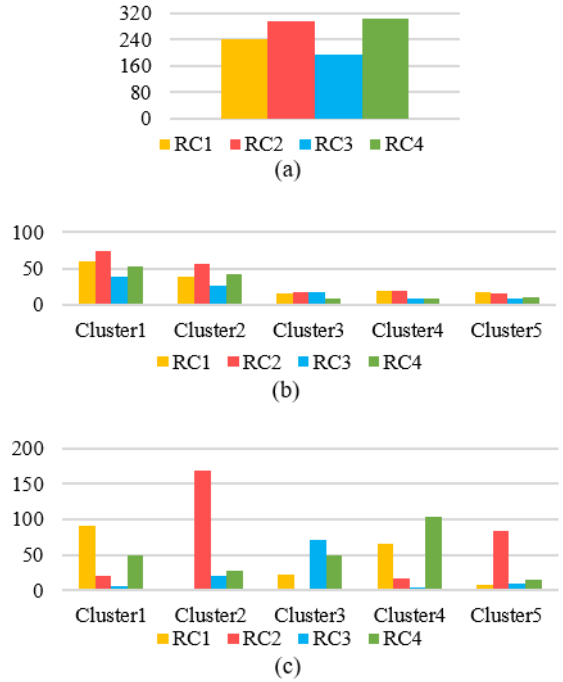

Fig. 9. Root-cause analysis results for the diagnosis system (Network Product #1): (a) the histogram of four root-causes labeled by human experts, and (b)-(c) the root causes associated with the five largest clusters identified by hierarchical clustering [7] and the proposed method respectively.

Fig. 9 further shows the root-cause analysis results for both methods. In this example, four root causes, denoted as RC1, RC2, RC3 and RC4 respectively, are labeled by human experts for the samples in ***S*** of the diagnosis system and their histogram is shown in Fig. 9(a). Fig. 9(b)-(c) show the root causes associated with the five largest clusters identified by hierarchical clustering [7] and the proposed method respectively. In Fig. 9(b), the four root causes are not clearly distinguished in each cluster, implying that the conventional method fails to cluster the defective samples of the diagnosis system based on their root causes. On the other hand, each cluster in Fig. 9(c) contains only 1~2 dominant root causes, thereby demonstrating the higher accuracy of the proposed method.

B. Network Product #2

In this example, 17 network system tests are applied to a network product with 6299 samples. These tests focus on checking the connectivity information (e.g., latency, package loss, etc.) of each sample. Human experts have helped to label

four possible root causes for this product as the ground truth. Among all 6299 samples, 3621 samples are labeled with the pass/fail information, and they form the set $\boldsymbol{S}_L$ with the historical test data $\boldsymbol{D}_L$ for training. No root-cause information is available for these 3621 samples. The other 6299 – 3621 = 2678 samples are defective, and they form the set $\boldsymbol{S}$ of the diagnosis system with the test data $\boldsymbol{D}$ in the diagnosis phase.

We utilize 5-fold cross validation is applied to select the effective depth T_{Dep} of the decision tree among 17 possible depth values {1, 2, …, 17}, given the fact that there are 17 test items in total. The classification accuracy $P_{Acc}(T_{Dep,d})$ computed using 5-fold cross validation is shown as a function of the depth of the decision tree $T_{Dep,d}$ in Fig. 10. As highlighted by the red cycle in Fig. 10, the effective depth is determined as $T_{Dep} = 4$, where we reach the maximum classification accuracy $P_{Acc}(T_{Dep}) = 0.89$.

Furthermore, the L-method is applied to select the effective threshold N_{Th} among 24 possible values {1%, 2%, 3%, …, 10%, 12.5%, 15%, 17.5%, 20%, 22.5%, 25%, 30%, 40%, …, 100%}×2678, considering the fact that 2678 defective samples exist in $\boldsymbol{S}_L$ for training. The intra-cluster distance $I_{Dis}(K_d)$ is shown as a function of the number of clusters K_d in Fig. 11. As highlighted by the red cycle in Fig. 11, the effective number of clusters is $K = 10$ based on the L-method. The corresponding effective threshold is determined as $N_{Th} = 7\% \times 2678 \approx 187$.

Fig. 12 shows the normalized mutual information I_{NMI} as a function of the number of clusters K_{HC} for the conventional hierarchical clustering method [7]. In this example, the optimal number of clusters is equal to 45, as highlighted by the red cycle in Fig. 12.

Based on these effective hyper-parameters: $T_{Dep} = 4$, $N_{Th} = 187$, $K = 10$ and $K_{HC} = 45$, we apply the conventional clustering method [7] and our proposed approach to the set $\boldsymbol{S}$ of the diagnosis system. In this example, our proposed decision tree incorrectly assigns 154 defective samples (out of 2678 defective samples) in $\boldsymbol{S}$ to the pass nodes and form the outlier cluster (i.e., "Stage-I Cluster 0" in Fig. 3).

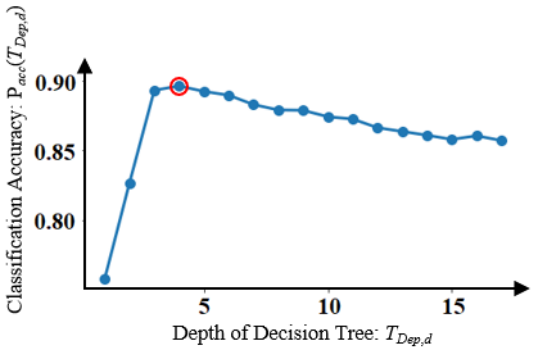

Fig. 10. Classification accuracy estimated by 5-fold cross validation as a function of the depth of decision tree for Network Product #2, where the effective depth is highlighted by the red cycle.

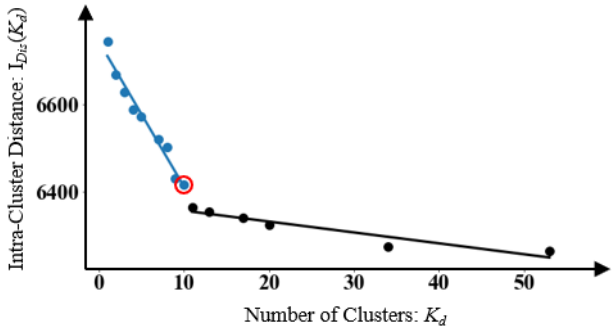

Fig. 11. Intra-cluster distance as a function of the number of clusters for Network Product #2, where the effective number of clusters, highlighted by the red cycle, is determined by the L-method.

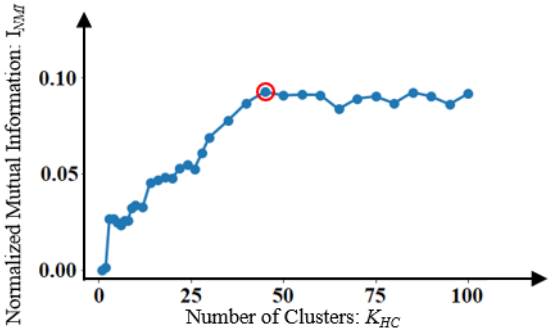

Fig. 12. Normalized mutual information as a function of the number of clusters for the conventional hierarchical clustering method [7] on Network Product #2, where the "best-yet-ideal" result is achieved with the optimal number of clusters as highlighted by the red cycle.

The efficiency of the conventional hierarchical clustering method [7] and the proposed method is compared for root-cause analysis in Table IV. Note that the proposed method outperforms the conventional hierarchical clustering method with substantially higher I_{NMI}, while the runtime for both methods is less than one minute.

More detailed results of root-cause analysis are shown in Fig. 13 for both methods. In this example, four root causes, denoted as RC1, RC2, RC3 and RC4 respectively, are labeled by human experts for the 2678 samples in $\boldsymbol{S}$ and their histogram is shown in Fig. 13(a). Fig. 13(b)-(c) show the histograms of root causes associated with the five largest clusters diagnosed by hierarchical clustering [7] and the proposed method respectively. In Fig. 13(b), the four root causes are not clearly distinguished in each cluster. In contrast, each cluster in Fig. 13(c) contains only 1~2 dominant root causes. These results, thereby, demonstrate that the proposed method achieves significantly better diagnosis performance than the conventional hierarchical clustering technique.

TABLE IV
PERFORMANCE OF ROOT-CAUSE ANALYSIS ON NETWORK PRODUCT #2

Method	I_{NMI}	K	Runtime (s)
Hierarchical clustering [7]	0.093	45	13.6
Proposed method	0.283	10	54.4

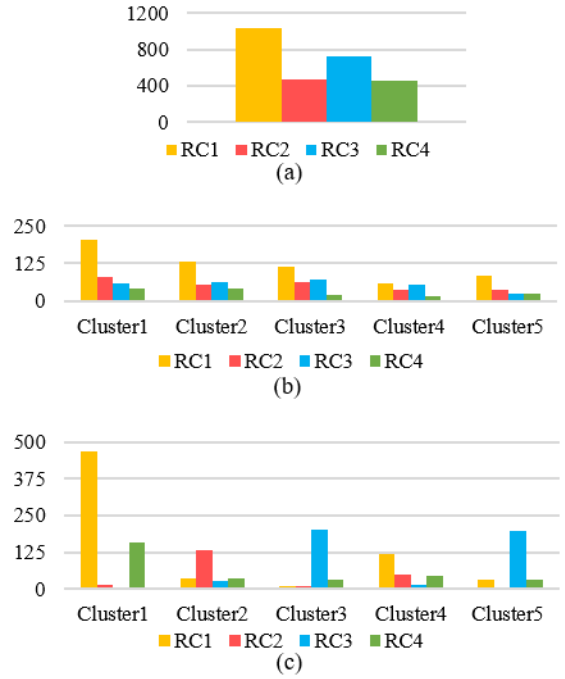

Fig. 13. Root-cause analysis results for the diagnosis system (Network Product #2): (a) the histogram of four root-causes labeled by human experts, and (b)-(c) the root causes associated with the five largest clusters identified by hierarchical clustering [7] and the proposed method respectively.

V. Conclusions

We have presented an unsupervised root-cause analysis method for integrated systems. First, we formulated the root-cause analysis as an unsupervised clustering problem. Next, a two-stage clustering method was presented to analyze the root causes. In the first stage, a decision-tree model leveraging the pass/fail information has been applied to roughly cluster the samples of a diagnosis system. In the second stage, frequent-pattern mining was exploited for precise clustering such that each cluster contained only a few dominant root causes. As demonstrated by two industry case studies, the proposed method has outperformed the state-of-the-art hierarchical clustering method and it is expected to offer useful insights to design/process engineers.

References

[1] M. Alioto, *Enabling the Internet of Things: From Integrated Circuits to Integrated Systems*, Springer, 2017.

[2] M. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel and L. Ladid, "Internet of things in the 5G era: enablers, architecture, and business models," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 510-527, 2016.

[3] T. Vo, Z. Wang, T. Eaton, P. Ghosh, H. Li, Y. Lee and W. Wang, "Design for board and system level structural test and diagnosis," *IEEE International Test Conference*, pp. 1-10, 2006.

[4] L. Wang, "Experience of data analytics in EDA and test—principles, promises, and challenges," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 6, pp. 885-898, 2016.

[5] H. Stratigopoulos, "Machine learning applications in IC testing," *IEEE European Test Symposium*, pp. 1-10, 2018.

[6] I. Elfadel, D. Boning and X. Li, *Machine Learning in VLSI Computer-Aided Design*, Springer, 2019.

[7] A. Gómez-Andrades, P. Muñoz, I. Serrano and R. Barco, "Automatic root cause analysis for LTE networks based on unsupervised techniques," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 2369-2386, 2015.

[8] Z. Zhang, Z. Wang, X. Gu and K. Chakrabarty, "Physical-defect modeling and optimization for fault-insertion test," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 20, no. 4, pp. 723-736, 2011.

[9] F. Ye, Z. Zhang, K. Chakrabarty and X. Gu, "Board-level functional fault diagnosis using artificial neural networks, support-vector machines, and weighted-majority voting," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 5, pp. 723-736, 2013.

[10] S.Jin, F. Ye, Z. Zhang, K. Chakrabarty and X. Gu, "Efficient board-level functional fault diagnosis with missing syndromes", *IEEE Transactions on Very Large Scale Integration Systems*, vol. 35, no. 6, pp. 985-998, 2015.

[11] F. Ye, Z. Zhang, K. Chakrabarty and X. Gu, "Adaptive board-level functional fault diagnosis using incremental decision trees," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vo. 35, no. 2, pp. 323-336, 2015.

[12] M. Liu, F. Ye, X. Li, K. Chakrabarty and X. Gu, "Board-level functional fault identification using streaming data," *IEEE VLSI Test Symposium*, pp.1-6, 2019.

[13] Y. Lei, F. Jia, J. Lin, S. Xing and S. Ding, "An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 5, pp. 3137-3147, 2016.

[14] F. Ye, Z. Zhang, K. Chakrabarty and X. Gu, "Knowledge discovery and knowledge transfer in board-level functional fault diagnosis," *IEEE International Test Conference*, pp. 1-10, 2014.

[15] J. Han, J. Pei, Y. Yin and R. Mao, "Mining frequent patterns without candidate generation: a frequent-pattern tree approach," *Data Mining and Knowledge Discovery*, vol. 8, no. 1, pp. 53-87, 2004.

[16] C. Manning, R. Prabhakar and S. Hinrich, *Introduction to Information Retrieval*, Cambridge University Press, 2009.

[17] D. Davies and D. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 2, pp. 224-227, 1979.

[18] A. Rosenberg and J. Hirschberg, "V-measure: a conditional entropy-based external cluster evaluation measure," *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 410-410, 2007.

[19] J. Han, H. Cheng, D. Xin and X. Yan, "Frequent pattern mining: current status and future directions," *Data Mining and Knowledge Discovery*, vol. 15, no. 1, pp. 55-86, 2007.

[20] B. Liu, Y. Xia and P. Yu, "Clustering through decision tree construction," *ACM International Conference on Information and Knowledge*, pp. 20-29, 2000.

[21] R. Srikant and R. Agrawal, "Mining quantitative association rules in large relational tables," *ACM SIGMOD Record*, vol. 25. no. 2, pp. 1-12, 1996.

[22] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[23] S. Salvador and P. Chan, "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms," *IEEE International Conference on Tools with Artificial Intelligence*, pp. 576-584, 2004.

[24] P. Tan, M. Steinbach and V. Kumar, *Introduction to Data Mining*, Pearson Education India, 2016.

[25] L. Breslow and D. Aha, "Simplifying decision trees: a survey," *The Knowledge Engineering Review*, vol. 12, no. 1, pp. 1-40, 1997.