RAPMiner: A Generic Anomaly Localization Mechanism for CDN System with Multi-dimensional KPIs

Chang Liu[†‡], Yanwei Liu[†*], Zhen Xu[†], Liang Dai[†‡]
[†]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[‡]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

***Abstract*—As essential work in IT operations, anomaly localization, aiming to identify the affected scope of Internet infrastructure once an anomaly alarm occurs, is challenging due to the huge search space. The existing solutions usually show limited performances in the CDN scenario since they take the desirable assumptions that do not match with the practical anomaly pattern features. To address this issue, in this paper, we propose RAPMiner, which first uses a classification power-based redundant attribute deletion to prune the non-root cause attribute combinations, and then adopts an anomaly confidence-guided layer-by-layer top-down search to avoid searching for anomaly but non-root patterns. Both of them are effective in narrowing the search space. Experimental results show that RAPMiner can achieve comparable performance with the SOTA approach on the published Squeeze dataset according to F1-score and efficiency, as well as the best RC@k with stable parameter sensitivity on the RAPMD.**

***Index Terms*—Anomaly localization, root anomaly pattern (RAP), cuboid, attribute combination**

I. Introduction

Due to the increasing complexity and growing scale of today's Internet infrastructures, failure on devices and service degradation are inevitable. To detect these failures and conduct service quality management, human operators measure and monitor various Key Performance Indicators (KPIs) [1]–[7]. Once an anomaly alarm occurs, it usually indicates the potential failure or service degradation for the system or Internet infrastructure, e.g., configuration errors, software defects, network and server machine overload or failures, etc., [1], [8]–[17]. To rapidly respond to these issues before they cause more critical performance degradation and guarantee the quality of experience (QoE) for users, anomaly localization is further needed in IT operations, i.e., to identify the affected scope of the specific system or infrastructure when it is triggered by an anomaly alarm and then switch the impacted users to the backup system timely.

As one of the most critical Internet infrastructures, Content Delivery Networks (CDNs) play a crucial role in delivering Internet content and guaranteeing high QoE [18]–[20]. With the growing scale, edge nodes of today's commercial CDN usually sink to districts or counties, providing services for hundreds or even thousands of websites in delivering content

*Yanwei Liu is the corresponding author

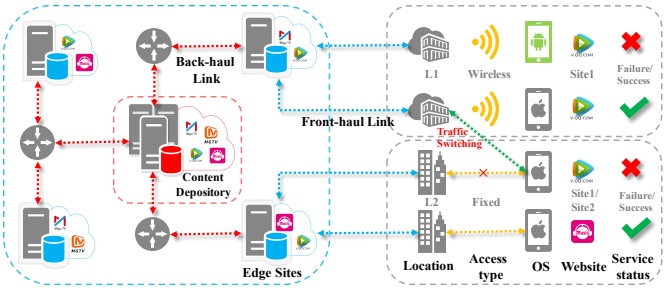


Fig. 1. IT operations in CDN Infrastructure

for various users. It is a great challenge to conduct IT operations for such a large distributed Internet infrastructure. The pioneering study [1] only focuses on anomaly detection for CDN to give an alarm, however, lacks anomaly localization to help identify the affected scope of the failure and provide human operators some clues for further trouble-shooting or root cause analysis. Via discussing with human operators, we are aware that the anomaly localization for the current CDN system mainly relies on manual work, i.e., mining the clues from massive data manually which is a very complex and time-consuming process. In this context, the need for an automatic method that conducts anomaly localization for CDN effectively and efficiently is urgent.

To this end, we start our work with a careful analysis of a real-world ISP-operated CDN in China. As shown in Fig. 1 right, users that are served by CDN can be characterized by a four-tuple or called attribute combination, i.e., the location of edge node, access type, operating system (OS) of the device, and the website that the user surfs. In practice, anomaly localization usually depends on KPIs with varying-grained attribute combinations. KPI for a fine-grained attribute combination usually indicates the performance or service status of the more specific scope generally, e.g., KPI for a four-tuple (L1, wireless, Android, Site1) represents the performance of "Android" users that try to request the contents from "Site1" and are served by the edge nodes at location "L1" via "wireless" network. While KPI for a coarse-grained attribute combination, e.g., (L1, *, *, Site1) indicates the service status of the users that request content from "Site1" and are served by the edge nodes at location "L1", regardless of the OS for

TABLE I
ATTRIBUTES OF CDN SYSTEM

Location (33)	Access Type (4)	OS (4)	Website (20)
L1	Wireless	Android	Site1
L2	Fixed	IOS	Site2
...	...	...	...

device and access type. The goal of anomaly localization is to find the attribute combination with the coarsest granularity among all of the attribute combinations whose KPIs present anomalous. Some studies call such attribute combination the root cause of the anomaly [21], [22] with its being the parent of all other attribute combinations with anomalous KPIs, or an effective combination of the emerging issue [14]. In this paper, we call it **R**oot **A**nomaly **P**attern (**RAP**). In the above example, comparing with (L1, wireless, Android, Site1), (L1, *, *, Site1) is more likely the RAP, since it describes more general characteristics among the affected scope.

Even though the existing studies have focused on mining root anomaly patterns for various Internet infrastructures or applications, however, the careful analysis still underscores the following two main challenges of mining root anomaly patterns for the real-world CDN.

Challenge 1: Huge Search Space. Table I shows some elements in each attribute for the real-world CDN that we studied, e.g., there are 33 specific elements in the attribute named "Location". Since the search space usually increases exponentially with the number of attributes and attribute elements as illustrated in [21], the search space in the CDN scenario will be up to $\mathbf{2^{20\times 33\times 4\times 4}}$. Consequently, such a huge space is a big challenge in designing an automatic algorithm to solve the infeasibility of manual localization of root anomaly patterns. The existing studies usually narrow the search space through some highly strict assumptions. For example, Adtributor [13] assumes that the root anomaly pattern is only a 1-dimensional attribute combination, and HotSpot [21] assumes that all of the root anomaly patterns are only located in a single cuboid. Both Squeeze [22] and HotSpot [21] assume the same anomaly magnitude for all attribute combinations under the same failure, i.e., the same relative deviation between the predicted value and the actual value of KPI. Besides, Squeeze also assumes varying anomaly magnitude for attribute combinations under different failures to combine clustering with searching. However, the careful analysis of the real-world ISP-operated CDN suggests that assumptions in existing methods are more desirable which limit their usefulness in the real-world CDN.

Challenge 2: Determination Rule for RAP. Obviously, the root anomaly patterns should exhibit the specific features that are apart from the non-root anomaly patterns. The challenge lies in how to characterize or describe such differences between the root and non-root anomaly patterns, i.e., how to determine the root anomaly patterns via a determination rule. Existing studies usually try to solve this issue by introducing the corresponding quantitative metrics for the specific scenarios. For example, HotSpot [21] introduces the ripple effect to characterize the magnitude relationship between the root cause attribute combinations and its "descendant" attribute combinations. Squeeze [22] further proposes a generic one, i.e., the generalized ripple effect for both fundamental and derived KPIs. Besides, iDice [14] identifies the effective combination via a metric called "Isolation Power", etc. Even though these quantitative metrics show effectiveness, however, some of them depend on the specific scenarios with corresponding assumptions. Hence, a proper determination rule is necessary for root anomaly patterns of the real-world CDN.

To cope with the limitations of previous work in tackling the two challenges above, in this paper, we propose a **R**oot **A**nomaly **P**attern **Miner** (**RAPMiner**), which considers the practical distribution features of root anomaly patterns to avoid the ideal assumptions, so as to be more practical without loss of effectiveness and efficiency. Specifically, RAPMiner first proposes a metric called Classification Power (CP) to determine whether an attribute is independent of a root anomaly pattern, and then combines it with the other metric called Anomaly Confidence (AC), which judges whether an attribute combination is anomalous or not, thus to address the *second challenge* above. Accordingly, RAPMiner solves the problem of mining root anomaly patterns in two stages: CP-based Redundant Attribute Deletion to prune the non-root cause attribute combinations and AC-guided Layer by Layer Top-down Search to avoid searching for anomaly but non-root patterns. Both two steps are effective in narrowing the search space, tackling the *first challenge* above. The contributions of this paper are summarized as follows.

- We propose RAPMiner, a framework for root anomaly patterns mining, which can localize root anomaly patterns effectively and efficiently without strict assumptions.
- We develop a highly efficient metric named classification power that can significantly reduce the search space by deleting redundant attributes. Our analysis demonstrates that even if only one attribute is pruned, the total search space will be narrowed by at least 50%.
- By injecting failures into the data collected from a large ISP-operated CDN in China, we create a semi-synthetic dataset and call it RAPMD. We refer to the real-world root anomaly patterns in failures injection procedures, so as to avoid the ideal root anomaly patterns in RAPMD.
- Experiments are conducted on two datasets including RAPMD to show the effectiveness of RAPMiner in CDN, and the semi-synthetic dataset published by Squeeze to illustrate its usefulness in other scenarios. The results show that RAPMiner achieves a comparable F1-score with the existing methods on the Squeeze dataset, and significantly outperforms the state-of-the-art approaches with regard to RC@k on RAPMD with stable parameters sensitivity. Besides, the running time experiments also show the efficiency of the proposed RAPMiner. For allowing experiment reproduction, we have released our source codes via GitHub at https://github.com/liuchang-sophie/RAPMiner.

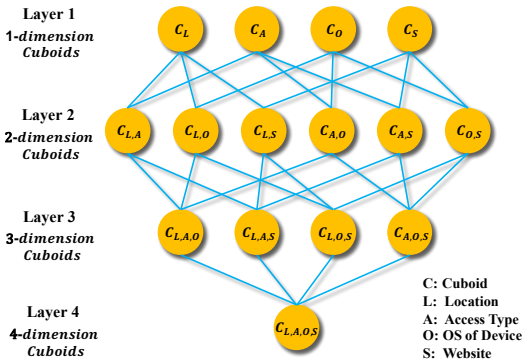


Fig. 2. The Hierarchical Structure of Cuboids

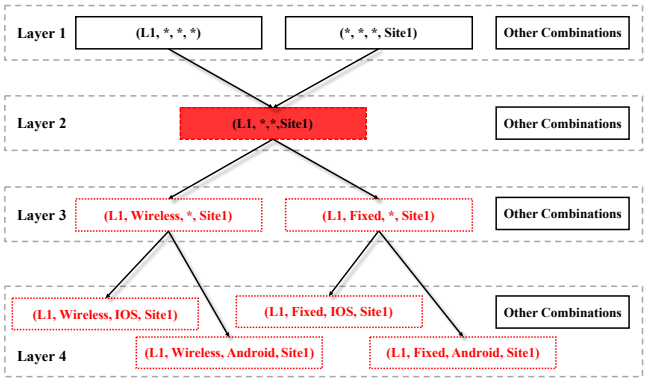


Fig. 3. The Root Anomaly Pattern

II. Background

In this section, we first introduce the IT operations in CDN infrastructure. Then, we will illustrate some details via examples, e.g., basic concepts, to further help understand the anomaly localization in CDN.

A. IT Operations in CDN

Fig. 1 left shows a typical CDN infrastructure. In CDN, the edge nodes cache the contents of various websites and usually sink to districts or counties which are very closed to users. When HTTP requests are sent by Internet users, the scheduling center gives priority to providing content to users from edge nodes via the front-haul link, or otherwise retrieving the contents from the upper depository firstly via the back-haul network, then sending them to users when a cache miss occurs.

To conduct service quality management for CDN, human operators usually collect and monitor various CDN KPIs, such as traffic volume, cache hit ratio and server response delay, etc. Once an anomaly alarm occurs, anomaly localization is triggered. As shown in Fig. 1 right, a four-tuple (L1, Wireless, Android, Site1) indicates the "Android" users surf the "Site1" and are served by the edge node at location "L1" via the "wireless" network, however, services failed, unfortunately. Compared these impacted users with those in a four-tuple (L1, Wireless, IOS, Site1) which are served successfully, it is not hard for us to observe that the "OS" attribute with the "Android" element may be the issue. Similarly, since the service for users that characterized by (L2, Fixed, IOS, Site1) is failed, while users belonging to (L2, Fixed, IOS, Site2) are served successfully, thus there is probably the failure of "Site1". In this case, clues provided by anomaly localization enable human operators timely to switch the impacted users to edge sites in "L1" to guarantee the QoE.

B. Analysis for Search Space

As mentioned above, the impacted scope of the ISP-operated CDN can be characterized by four attributes, i.e. location, access type, OS of the device and website user surfs. Each attribute has various elements which can form many different attribute combinations. Consistent with HotSpot [21], we use **cuboids** to divide these attribute combinations. As shown in Fig. 2, there are 15 cuboids in the 4-attribute CDN system with the generalized form 2^n-1 where n is the number of attributes. These cuboids locate in four layers, which are 1-dimensional cuboids with only one attribute, 2-dimensional cuboids with two attributes, 3-dimensional cuboids with three attributes, and 4-dimensional cuboids with four attributes respectively. There is an obvious parent-child relationship between the layers. For example, cuboid C_L is only composed of Location and contains only 33 attribute combinations, while cuboid $C_{L,S}$ is composed of Location and Website which contains $660(=20\times33)$ attribute combinations. The worst case is that cuboid $C_{L,A,O,S}$ is composed of all attributes, and the number of attribute combinations it contains has reached $10560(=20\times33\times4\times4)$.

C. Anomaly Localization in CDN: an Example Analysis

When a failure alarm occurs, the overall KPI of the CDN usually shows abnormal behaviors. There may be an anomaly in the KPI curves of many attribute combinations in all cuboids shown in Fig. 2, but actually not every attribute combination showing abnormal behavior will be called the root anomaly pattern. This is because of the inclusion and parent-child relationship between attribute combinations, i.e., the affected scope covered by the higher attribute combination contains the scope characterized by the lower attribute combination and only those attribute combinations that present anomalies and no longer have the parents presenting anomalies can be called root anomaly patterns. As shown in Fig. 3, the KPI curves of attribute combinations (L1, *, *, Site1), (L1, Wireless, *, Site1), (L1, Fixed, *, Site1), (L1, Wireless, IOS, Site1), (L1, Wireless, Android, Site1), (L1, Fixed, IOS, Site1) and (L1, Fixed, Android, Site1) all show abnormal behaviors when an anomalous alarm occurs. However, since (L1, *, *, Site1) is the ancestor of all other anomalous attribute combinations, covering the impacted scope of all other anomalous attribute combinations, hence it is the RAP we want to identify.

III. Problem Definition

Before introducing the problem definition, we define some of the notations in Table II which will be used later.

A. KPI

KPIs collected in real-world Internet infrastructures are mainly classified into two categories. One category is the

TABLE II
SUMMARY OF NOTATIONS

Notations	Definations
ac	Attribute Combination
$attr$	Attribute
$Cub_{AttributeSet}$	Cuboid consisting of attribute combinations which have the same attributes
v	Actual Value
f	Forecast Value
D	Basic Dataset, i.e. $Cub_{\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D}}$
D_{attr_i}	Sub-dataset which is produced from the basic dataset D and contains $attr_i$
CP_{attr}	The classified power of the specific attribute
t_{CP}	The threshold of classification power to determine whether some attribute is redundant, in the form of percentage
t_{conf}	The threshold of confidence to determine whether some attribute combination is anomalous, in the range of $(0, 1)$
$Layer$	The index of the layer that some attribute combination locates in
$Parents()$	The function to get the parent set of some attribute combination
$Elem()$	The function to get elements of some attribute
$l()$	The function to get the number of elements for each attribute
$AttributeSet()$	The function to get the attribute set of some attribute combinations
$Info()$	The function to get the information entropy of the dataset according to Shannon Theory
$Info_{attr}()$	The function to get the sum of information entropy of all of the datasets which are formed by dividing the dataset D with some attribute
$Confidence()$	The function to get the confidence
$support\_count_D()$	The function to get the support on dataset D
$Descendants()$	The function to get all of the descendants of some attribute combination

fundamental KPI, and the other is the derived KPI [13]. In the scenario of ISP-operated CDN, human operators usually collect and aggregate the fundamental KPIs of the most fine-grained attribute combinations once an anomaly alarm is triggered. Since the fundamental KPIs are generally additive, the KPIs of the coarse-grained attribute combinations in the higher layers can be obtained from the KPIs of the fine-grained attribute combinations in the lower layers, as shown in Fig. 4. Even though the derived KPIs are non-additive, we can obtain them from fundamental KPIs through a series of transformations, i.e. $K^D = g(K_1^F, K_2^F, ..., K_m^F)$, where K^F is fundamental KPI and K^D is derived KPI, and g is the specific functions. Therefore, as shown in Fig. 4 we can first obtain the fundamental KPIs of the coarse-grained attribute combinations in higher layers via the aggregation from the lower layers and further get the derived KPIs through the specific transformations.

B. Root Anomaly Pattern (RAP)

Based on the description of the root anomaly pattern in Section II, we now define it precisely. We denote the attribute combination as ac.

Definition 1 (Root Anomaly Pattern, RAP): If ac is an anomalous attribute combination, and $\nexists ac' \in Parents(ac)$ is an anomalous attribute combination, then ac is a RAP.

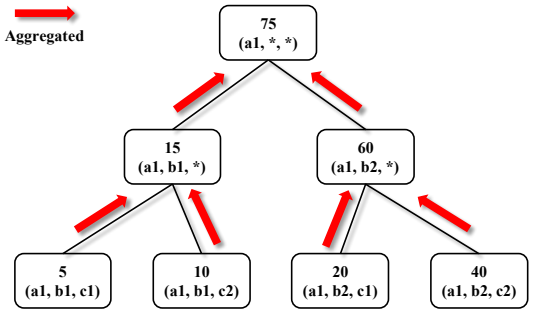


Fig. 4. The Aggregated Process of Fundamental KPI

C. Problem Definition

For the sake of the description, we still use a four-tuple attribute combination to elaborate the problem definition. Note that the number of attributes can be generalized to any value. We denote the attribute set for the impacted scope $\mathcal{S}$ as $AttributeSet(\mathcal{S}) = \{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}\}$, and the element sets for these attributes are denoted as $Elem(\mathcal{A}) = \{a_1, a_2, ..., a_{l(\mathcal{A})}\}$, $Elem(\mathcal{B}) = \{b_1, b_2, ..., b_{l(\mathcal{B})}\}$, $Elem(\mathcal{C}) = \{c_1, c_2, ..., c_{l(\mathcal{C})}\}$ and $Elem(\mathcal{D}) = \{d_1, d_2, ..., d_{l(\mathcal{D})}\}$ respectively, where $l(\mathcal{A})$, $l(\mathcal{B})$, $l(\mathcal{C})$ and $l(\mathcal{D})$ indicate the number of elements for each attribute. Then, we divide the cuboids composed of these four attributes into four layers. Where cuboids in the first layer only contain an individual attribute, and cuboids in the second layer can be obtained by the Cartesian product of any two attributes, so as the third and the fourth layers, as shown in Fig. 2. We formulate the cuboid as follows and only show the example of one cuboid for each layer.

$$
\begin{aligned}
&Cub_{\mathcal{A}} = \{(a_1, *, *, *), ..., (a_{l(\mathcal{A})}, *, *, *)\},\\
&length = l(\mathcal{A}).\\
&Cub_{\mathcal{A},\mathcal{B}} = \{(a_1, b_1, *, *), ..., (a_{l(\mathcal{A})}, b_{l(\mathcal{B})}, *, *)\},\\
&length = l(\mathcal{A}) \times l(\mathcal{B}).\\
&Cub_{\mathcal{A},\mathcal{B},\mathcal{C}} = \{(a_1, b_1, c_1, *), ..., (a_{l(\mathcal{A})}, b_{l(\mathcal{B})}, c_{l(\mathcal{C})}, *)\},\\
&length = l(\mathcal{A}) \times l(\mathcal{B}) \times l(\mathcal{C}).\\
&Cub_{\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D}} = \{(a_1, b_1, c_1, d_1), ..., (a_{l(\mathcal{A})}, b_{l(\mathcal{B})}, c_{l(\mathcal{C})}, d_{l(\mathcal{D})})\},\\
&length = l(\mathcal{A}) \times l(\mathcal{B}) \times l(\mathcal{C}) \times l(\mathcal{D}).
\end{aligned}
$$

As for the impacted scope characterized in the four-tuple, $Cub_{\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D}}$ is the set of the most fine-grained attribute combinations. Each element in this set is a leaf attribute combination. They only have ancestors but no descendants. The real value of KPIs for each element in the $Cub_{\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D}}$ can be obtained in the data collection phase, and we can get the corresponding predicted values via some prediction methods. We denote the actual and predicted value as v and f, respectively. Examples of the data are shown in Table III. Since many existing studies have focused on prediction techniques [1], [2], [15], [23], [24], we do not take the prediction methods as our primary work in this paper.

To sum up, mining of root anomaly patterns can be expressed as: given the actual value v and predicted value f of the corresponding KPIs for the most fine-grained attribute combination set $Cub_{\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D}}$, how we can quickly and accurately mine the $RAPs$ of the impacted scope $\mathcal{S}$.

TABLE III
THE DATA FORM OF THE MOST FINE-GRAINED ATTRIBUTE COMBINATION

Attribute Combination	v	f
(a_1, b_1, c_1, d_1)	10.0	5.0
(a_1, b_1, c_1, d_2)	23.0	20.5
...	...	...
$(a_{l(\mathcal{A})}, b_{l(\mathcal{B})}, c_{l(\mathcal{C})}, d_{l(\mathcal{D})})$	101.2	125.8

IV. ROOT ANOMALY PATTERN MINER

A. Insights

Through a careful analysis of the real-world CDN dataset, we underscore two domain-specific insights as follows.

1) Insight 1: If an attribute is in the root anomaly patterns, this attribute has some degree of classification power for the entire most fine-grained attribute combinations, namely normal or abnormal.

2) Insight 2: If an attribute combination is a root anomaly pattern, most of its children should be anomalous, while all of its parents should not be anomalous. In other words, if only a small part of its child attribute combinations are anomalous or anyone parent is anomalous, such attribute combination should not be considered as a RAP.

B. Framework

Fig. 5 shows the overall framework of RAPMiner, which mainly consists of two parts: a CP-based Redundant Attribute Deletion and an AC-guided Layer by Layer Top-down Search. Specifically, the classification power-based redundant attribute deletion is designed for pruning the non-root cause attribute combinations. As illustrated in Section I, even if only one attribute is deleted, it will narrow about 50% of the search space. Besides, we also adopt Anomaly Confidence-guided Layer by Layer Top-down Search to avoid searching for anomaly but non-root patterns. Then, RAPMiner can traverse the search space layer by layer from top to bottom based on Breadth-First Search (BFS). The intuition is that if the currently searched attribute combination is anomalous and is considered as a RAP, the child attribute combinations are certainly not RAPs and can be pruned off directly. The input of RAPMiner is the anomaly detection results for KPIs of the most fine-grained attribute combinations. The existing studies usually use different localizing strategies for fundamental KPIs and derived KPIs, especially in the design of root cause scores. However, there is no need to differentiate KPIs in RAPMiner, because RAPMiner only uses the anomaly detection results for the most fine-grained attribute combinations and does not care about whether they are fundamental or derived KPIs. Since the anomaly detection for the most fine-grained attribute combinations can be done directly in the data collection stage without subsequent aggregation, the method is more general and helps save the data aggregation time.

C. Classification Power Based Redundant Attribute Deletion

It can be observed from Insight 1 that if an attribute is in the root anomaly patterns, using it to divide the dataset composed of the most fine-grained attribute combinations will reduce

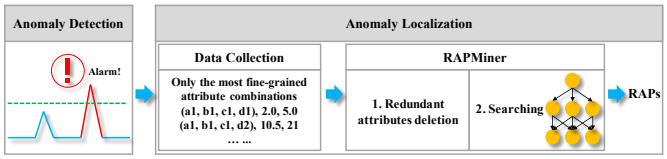


Fig. 5. The Framework of RAPMiner

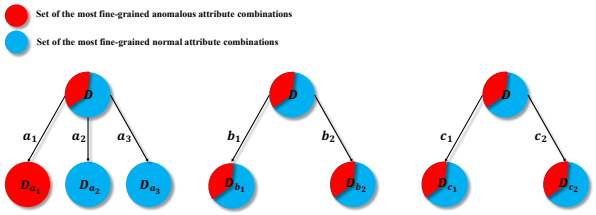


Fig. 6. Attributes Classifying the Most Fine-grained Attribute Combinations

information entropy [25]–[27]. When an attribute is selected to classify the dataset, i.e., divide the dataset into several sets according to the specific element in the attribute, as shown in Fig. 6, if it can hardly reduce the information entropy of the whole dataset, it is considered that the attribute has almost no ability to classify the dataset intuitively. Thus it cannot be an attribute in the root anomaly patterns. In other words, such an attribute is redundant and has nothing to do with the anomalies. In this way, we can delete redundant attributes before searching root anomaly patterns based on Insight 1 to narrow the search space and improve the efficiency in localizing the root anomaly patterns. Actually, there are many redundant attributes when a failure occurs in the real-world system. We also observe that the root anomaly patterns are mostly in the higher-layer cuboids in Fig. 2. We will later show that removing redundant attributes may dramatically improve the efficiency of the latter search process.

Fig. 6 shows a simple example. Let D be a dataset composed of all the most fine-grained attribute combinations of attributes $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$. The elements in attribute $\mathcal{A}$ are a_1, a_2 and a_3, the elements in attribute $\mathcal{B}$ are b_1, b_2, and the elements in attribute $\mathcal{C}$ are c_1, c_2. We assume that $(a_1, *, *)$ is the RAP. If we use attribute $\mathcal{A}$ to divide the dataset D, D tends to be more orderly, due to the fact that all of the most fine-grained anomalous attribute combinations related to a_1 can be classified into the anomalous set D_{a_1} and the rest of sets D_{a_2} and D_{a_3} may only contain the normal attribute combinations, as shown in Fig. 6 left. Thus the overall dataset entropy after the partition will be smaller than that when using attribute $\mathcal{B}$ or $\mathcal{C}$ to divide D. On the contrary, as shown in Fig. 6 middle, both the anomalous and normal attribute combinations are likely to be divided into the sets D_{b_1} and D_{b_2} simultaneously when we use attribute $\mathcal{B}$ to separate D. Note that the anomalous parts in D_{b_1} and D_{b_2} are derived from the most fine-grained anomalous attribute combinations that related to a_1. So similar results when we apply attribute $\mathcal{C}$ to classify D.

To sum up, we introduce the Classification Power (CP) to indicate how likely an attribute will be in RAPs by quantifying the entropy reduction when applied to divide the dataset of the

most fine-grained attribute combinations. The bigger the CP is, the more likely the attribute is to be in $RAPs$. Specifically, we formulate CP as:

$$CP_{attr} = \frac{Info(D) - Info_{attr}(D)}{Info(D)} \tag{1a}$$

$$Info(D) = -(p_a log p_a + p_n log p_n) \tag{1b}$$

$$Info_{attr}(D) = -\sum_{i=1}^{n} \frac{|D_{attr_i}|}{|D|}(p_a^{attr_i} log p_a^{attr_i} + p_n^{attr_i} log p_n^{attr_i}) \tag{1c}$$

where, p_a denotes the probability of abnormal attribute combinations in the dataset, while p_n denotes the probability of normal ones in the dataset. $p_a^{attr_i}$ is the probability of anomalous attribute combinations in attribute $attr_i$ branch, while $p_n^{attr_i}$ indicates the probability of normal ones in $attr_i$ branch. Based on Eq.1, we define a Criteria to determine whether an attribute is redundant or not below.

Criteria 1: $\forall attr \in AttributeSet(RAPs)$, $CP_{attr} > t_{CP}$, t_{CP} is a threshold with a very small value. Otherwise when $CP_{attr} \leq t_{CP}$, $attr \notin AttributeSet(RAPs)$.

Note that t_{CP} is a threshold in the form of percentage. The smaller the selected t_{CP}, the more strict standard for redundant attributes judgment, i.e., the classification power of an attribute must be extremely small before it can be considered as redundant, and thus the fewer redundant attributes can be deleted. On the contrary, when the selected t_{CP} is larger, it indicates that the judgment standard of redundant attributes is more relaxed, i.e., an attribute can be considered redundant if its classification power is less than a relatively small value. In this way, more redundant attributes can be deleted to improve the efficiency, with the sacrifice of accuracy.

Based on Criteria 1, we can remove all redundant attributes before searching for the root anomaly patterns. Proof 1 demonstrates that deleting redundant attributes can significantly narrow the search space. In addition, Table IV shows some details, e.g., deleting one redundant attribute will result in at least a 50% decrease of cuboids we have to search. Deleting two redundant attributes can reduce the cuboids by more than 75%, etc. The more redundant attributes we delete, the fewer cuboids need to be searched, thus resulting in the higher efficiency of localizing the root anomaly patterns. The complete procedure of redundant attribute deletion is given in Algorithm 1.

Proof 1: Denote the total number of attributes as n, and the total number of cuboids that need to be traversed when searching the root anomaly patterns is $2^n - 1$ without deleting any attributes. Assuming that k redundant attributes are deleted, the total number of cuboids to be traversed in the remaining search are $2^{n-k} - 1$, so the ratio of cuboids decreased to be traversed is:

$$\begin{aligned} DecreaseRatio@k =& \frac{(2^n - 1) - (2^{n-k} - 1)}{2^n - 1} = \frac{2^n - 2^{n-k}}{2^n - 1} \\ =& \frac{2^k - 1}{2^k - \frac{1}{2^{n-k}}} > \frac{2^k - 1}{2^k} \end{aligned} \tag{2}$$

TABLE IV
THE RATIO OF CUBOIDS DECREASED AFTER DELETING REDUNDANT ATTRIBUTES

k	1	2	3	4	5
$DecreaseRatio@k$	0.5	0.75	0.875	0.9375	0.96875

Algorithm 1 Redundant Attributes Deletion

Input: The most fine-grained attribute combinations dataset D, each item of which has been detected as normal or abnormal by predicted value and real value, e.g., $[[a_1, b_1, c_1, d_1, anomalous], [a_2, b_2, c_2, d_2, normal], ...]$
The attributes set $AttributeSet$, e.g., $\{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}\}$
The threshold t_{CP}

Output: The left attributes set $AttributeSet'$, any of which is related with $RAPs$, e.g., $\{\mathcal{A}, \mathcal{B}\}$

```
for attr ∈ AttributeSet do
    Calculate CP_attr (Eq.1) on D
    if CP_attr < t_CP then
        delete attr from AttributeSet
    end if
end for
AttributeSet' ← Sort AttributeSet by CP_attr reversely
return AttributeSet'
```

TABLE V
MAPPING BETWEEN VERTICES AND ATTRIBUTE COMBINATIONS

1-1	$(a_1, *, *, *)$	2-6	$(a_2, b_2, *, *)$	3-2	$(a_1, b_1, c_2, *)$
1-2	$(a_2, *, *, *)$	2-7	$(a_2, *, c_1, *)$	3-3	$(a_1, b_2, c_1, *)$
1-3	$(a_3, *, *, *)$	2-8	$(a_2, *, c_2, *)$	3-4	$(a_1, b_2, c_2, *)$
1-4	$(*, b_1, *, *)$	2-9	$(a_3, b_1, *, *)$	3-5	$(a_2, b_1, c_1, *)$
1-5	$(*, b_2, *, *)$	2-10	$(a_3, b_2, *, *)$	3-6	$(a_2, b_1, c_2, *)$
1-6	$(*, *, c_1, *)$	2-11	$(a_3, *, c_1, *)$	3-7	$(a_2, b_2, c_1, *)$
1-7	$(*, *, c_2, *)$	2-12	$(a_3, *, c_2, *)$	3-8	$(a_2, b_2, c_2, *)$
2-1	$(a_1, b_1, *, *)$	2-13	$(*, b_1, c_1, *)$	3-9	$(a_3, b_1, c_1, *)$
2-2	$(a_1, b_2, *, *)$	2-14	$(*, b_1, c_2, *)$	3-10	$(a_3, b_1, c_2, *)$
2-3	$(a_1, *, c_1, *)$	2-15	$(*, b_2, c_1, *)$	3-11	$(a_3, b_2, c_1, *)$
2-4	$(a_1, *, c_2, *)$	2-16	$(*, b_2, c_2, *)$	3-12	$(a_3, b_2, c_2, *)$
2-5	$(a_2, b_1, *, *)$	3-1	$(a_1, b_1, c_1, *)$	-	-

D. Anomaly Confidence Guided Layer-by-Layer Top-down Search

After deleting the redundant attributes unrelated to the anomaly, we need to traverse all the cuboids composed of the remaining attributes to find the root anomaly patterns because they are all related to the root anomaly patterns. Based on Insight 2, once an attribute combination is a root anomaly pattern, its descendants can no longer be root anomaly patterns. Therefore, we design a layer-by-layer top-down search algorithm guided by the "Anomaly Confidence" metric, to localize the root anomaly patterns for failures as accurately and quickly as possible. The basic idea of the search algorithm is: we first use the "anomaly confidence" metric to determine whether an attribute combination is anomalous or not; Then, once an anomalous attribute combination is further considered as a candidate root anomaly pattern, all of its descendants can be pruned off.

Before going into the details, we first present Criteria 2 and Criteria 3.

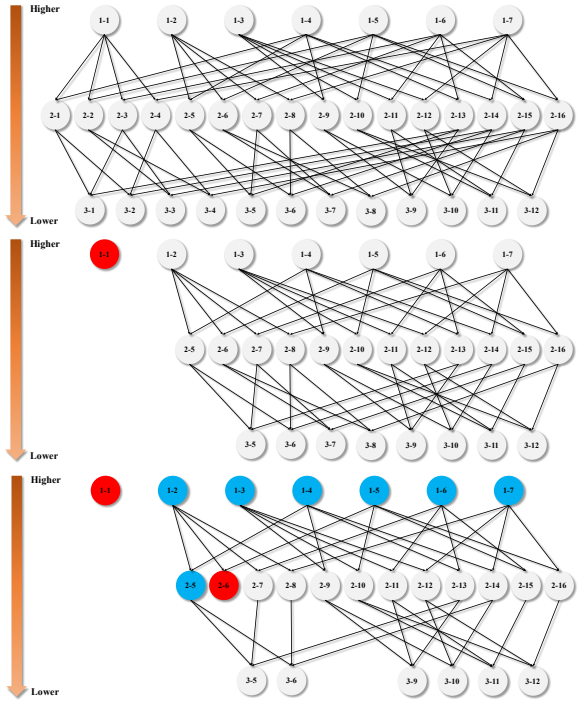


Fig. 7. Attribute Combinations Structure

Criteria 2: $\forall ac$, if $Confidence(ac \Rightarrow Anomaly) > t_{conf}$, then ac is anomalous. Where $Confidence(ac \Rightarrow Anomaly) = \frac{support\_count_D(ac, Anomaly)}{support\_count_D(ac)}$, t_{conf} is confidence threshold with a large value.

Criteria 3: If ac is a RAP, $\forall ac' \in Descendants(ac)$ is certainly not a RAP, where ac denotes the current searching attribute combination and D is the most fine-grained dataset.

In Criteria 2, t_{conf} is a confidence threshold with a large value. $Confidence(ac \Rightarrow Anomaly)$ also means the anomaly ratio of the current visiting attribute combination, which is expressed by the percentage of anomalies in the most fine-grained descendant attribute combinations of the current ac. $support\_count_D(ac)$ is the number of the descendant attribute combinations of the current visiting ac in the most fine-grained attribute combination dataset D, whereas $support\_count_D(ac, Anomaly)$ is the number of attribute combinations which are the descendants of ac and meanwhile anomalous in D. We can easily come to that the higher the confidence, the more likely the current ac is anomalous. Therefore, we should choose a relatively large t_{conf} instead of a very large t_{conf}, due to the fact that a relatively large t_{conf} will achieve a good error-tolerant rate.

Subsequently, we present the detailed explanation below combined with the directed acyclic graph (DAG) in Fig. 7. In the DAG, each vertex represents an attribute combination, and each edge indicates the parent-child relationship between two vertices. Note that the start vertex of the edge is the parent of the end vertex. Fig. 7 top shows the parent-child relationship of all attribute combinations before search. We assume that the root anomaly patterns are $(a_1, *, *, *)$ and $(a_2, b_2, *, *)$. Next, we traverse each vertex layer-by-layer from top to bottom based on BFS. As shown in Fig. 7 middle, when we search the first vertex $(a_1, *, *, *)$ (node "1-1"), we can conclude that it is an anomalous attribute combination according to Criteria 2, and then according to the definition of RAP, it can be determined as a RAP. Meanwhile, all of its descendants are certainly not RAPs according to Criteria 3, which can be pruned off to avoid unnecessary search, thus improving efficiency. As shown in Fig. 7 bottom, after traversing the remaining vertices of the first layer based on BFS and finding no anomalous vertices according to Criteria 2, we start traversing the vertices of the second layer in the DAG. When traversing the vertex "2-6", we can conclude that it is a RAP according to Definition 1 and Criteria 2, so we delete all of its descendant vertices according to Criteria 3.

Next, we use an early stop strategy to terminate the search procedure of the DAG in advance, i.e., whenever we find a RAP, judge whether the current RAP candidate set has covered the most fine-grained anomalous attribute combination in the dataset D. Finally, we sort the obtained candidate set in descending order according to their RAPScore, (Eq. 3) and return top-k RAPs. We define the RAPScore by considering the layer that ac resides in, because the possibility that the current ac is a root cause is negatively correlated with it.

$$RAPScore = \frac{Confidence(ac \Rightarrow Anomaly)}{\sqrt{Layer}} \tag{3}$$

The detailed procedure of layer-by-layer top-down search is shown in Algorithm 2. Besides, it also should be noted that, in Fig. 7, red vertices denote the abnormal attribute combinations, blue vertices are normal attribute combinations that have been searched, and white vertices denote attribute combinations that have not been searched. The mapping between vertices and attribute combinations is reported in Table V.

V. Experiments

A. Dataset

We use two datasets, the public Squeeze semi-synthetic dataset [22] and the semi-synthetic dataset RAPMD. RAPMD is created by injecting failures into the background data which is collected from an ISP-operated CDN in China. Specifically, the background data consists of KPIs of the most fine-grained attribute combination, e.g., "Out_Flow" of (L1, Wireless, IOS, Site1), which indicates the traffic volume output from the edge servers at location "L1" for the "IOS" users who surf the "Site1" via the "wireless" network. And these KPIs are collected every 60 seconds spanning about 35 days (from Feburary 1st to March 7th). There are about 1440 time points every day, of which 3 time points are randomly selected for failures injection, thus we obtain 105 failures in total.

For the Squeeze dataset, it makes two assumptions.

1) Vertical Assumption: The anomaly degree of descendant attribute combinations under the same RAP is the same.

Algorithm 2 AC-Guided Layer-by-Layer Top-down Search

Input: The most fine-grained attribute combinations dataset D, the same as Alg.1
The attributes set $AttributeSet'$ returned by Alg.1, e.g., $\{\mathcal{A}, \mathcal{B}\}$
The threshold t_{conf}
The specified number of returned RAPs k
Output: The Root Anomaly Patterns Set, short for $RAPSet$, e.g., $[(a_1, *, *, *), (a_2, b_2, *, *)]$

for $Layer \in 1, 2, ..., |AttributeSet'|$ **do**
  $Cuboids \leftarrow get\_cuboids(AttributeSet', Layer)$
  for $cuboid \in Cuboids$ **do**
    for $ac \in cuboid$ **do**
      Calculate $Confidence(ac \Rightarrow Anomaly)$ on D
      if $Confidence(ac \Rightarrow Anomaly) > t_{conf}$ **then**
        $CandidateSet \leftarrow CandidateSet + ac$
        Prune off the brunch of ac
        if $CandidateSet$ Covers $D_{anomalous}$ **then**
          Early stop and jump out of all loops.
        end if
      end if
    end for
  end for
end for
for $CandidateRAP \in CandidateSet$ **do**
  Calculate $RAPScore$ (Eq.3) of $CandidateRAP$
end for
$CandidateSet' \leftarrow$Sort $CandidateSet$ by $RAPScore$ reversely
$RAPSet \leftarrow CandidateSet'(k)$
return $RAPSet$

2) Horizontal Assumption: The anomaly degree of different failures is different. Note that there may be several RAPs with regard to one failure.

However, through a careful analysis of the failure data of an ISP-operated CDN, we observe that even for the descendant attribute combinations under the same RAP, their anomaly magnitude may not be the same. Due to the fact that the KPI for the most fine-grained attribute combination in the real-world CDN is usually sparse and fails to show the statistical characteristic, resulting in the non-concentrated deviation between the predicted and actual values. On the contrary, we also observe that two anomalies may show a similar degree, even though they come from different failures in real-world CDN. In addition, the Squeeze dataset can be classified into different categories with regard to the dimension and the number of RAPs at a certain time point. However, it is usually very hard for us to know how many failures there are and which layer of cuboids the failure in the real-world scenario.

To make up for the above three shortcomings of the squeeze semi-synthetic dataset, we create RAPMD by referring to the real-world root anomaly patterns in failures injection procedure to avoid the ideal root anomaly patterns in RAPMD. To this end, we randomly extract 105-timestep points from the background data of a real-world ISP-operated CDN and inject the failures with the following characteristics.

- Randomness 1: We randomly select the number of RAPs that is in the range of $[1, 3]$ at each time point. Any dimension can be selected for each RAP, and the dimension between them is not necessary to be the same. For example, for time point 001, we randomly select three RAPs to inject faults, and these three RAPs are randomly selected as $(a_1, *, *, *)$,$(*, b_1, *, *)$ and $(*, *, c_1, d_1)$.
- Randomness 2: For each randomly selected root RAP's most fine-grained descendant attribute combination, we randomly select its Dev at $[0.1, 0.9]$ (Eq. 4, where ε is an extremely small value used to avoid dividing by 0). For each of the remaining normal most fine-grained attribute combinations, we randomly select Dev at $[-0.02, 0.09]$. Finally, the predicted value (Eq. 5) for each of the most fine-grained attribute combinations is given based on the selected Dev. This ensures that the relative deviation of the most fine-grained attribute combinations under the same RAP may be different, and the relative deviation of the most fine-grained attribute combination under different RAPs may be the same.

$$Dev = \frac{f - v}{f + \varepsilon} \tag{4}$$

$$f = \frac{v + Dev \cdot \varepsilon}{1 - Dev} \tag{5}$$

B. Evaluation Metrics

Because the Squeeze semi-synthetic dataset is classified according to the dimension and number of root anomaly patterns, we know the number of RAPs in advance before localizing. Therefore, we keep the number of returned results of the algorithm the same as the actual number of RAPs, and compare $F1$-score as Eq. 6 of baseline methods and RAPMiner in the experiments.

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6}$$

Since the number of failures injected into RAPMD at each time point is random, and we pay more attention to the recall, we adopt $RC@k$ [28]–[30] to evaluate the performance of baseline methods and RAPMiner, Eq. 7, where k is the number of recommended results of the algorithm and T is the anomaly set. $Pred_t^i$ denotes the i^{th} recommended result of the anomaly t, and $Real_t$ denotes the real RAP set of the anomaly t.

$$RC@k = \frac{\sum_{t \in T} \sum_{i=1}^{k} Pred_t^i \in Real_t}{\sum_{t \in T} |Real_t|} \tag{7}$$

To evaluate the efficiency of these algorithms, we directly compare their average running time in identifying the RAPs.

C. Baseline Methods

1) Adtributor [13]: This method is only applicable to the scenario where the RAPs are all one-dimensional. Adtributor proposes three metrics which are Explanation Power (EP), Succinctness and Surprise respectively to search each attribute value under each attribute and determine whether the attribute value is the culprit of failure or anomaly.

2) iDice [14]: iDice is able to conduct root anomaly pattern mining in multi-dimensional attribute combination space. It mainly uses Impact for data pruning, Change Detection and Isolation Power for localizing failures, i.e. mining RAPs based on BFS.

3) FP-growth [15]: Association rule is employed to search for root anomaly patterns. We adopt an efficient implementation of the association rule, i.e., FP-growth [31], [32] to mine the RAPs.

4) Squeeze [22]: A metric named GPS (General Potential Score) is proposed to determine the root anomaly pattern and tries to mine the root anomaly patterns by searching every possible cuboid based on clustering. As mentioned above, this method has two strict assumptions: (1) The anomaly magnitude of all attribute combinations under the same root anomaly pattern is the same; (2) The anomaly magnitude of attribute combinations varies among different failures.

D. Hardware platform

Our experiments, including performance and running time comparison, are conducted on a server with Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz.

E. Effectiveness Comparison

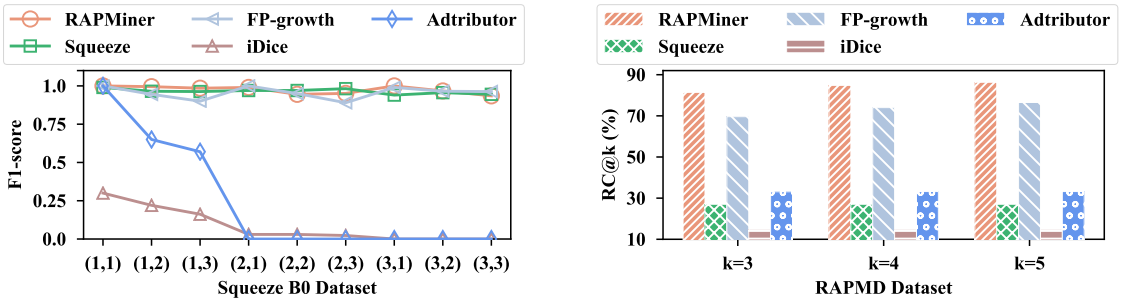
1) The Effectiveness on Squeeze Dataset: The Squeeze dataset has several groups of data according to different noise levels. We only select data of B0-level to evaluate the effectiveness of the methods because the varying noise levels only affect the anomaly detection of each most fine-grained attribute combination. However, RAPMiner does not focus on predicting and detecting KPI for each most fine-grained attribute combination. The subsequent *RAPScore* no longer uses the predicted and actual value of KPI but directly uses the anomaly detection results of each most fine-grained attribute combination for anomaly localization. The more accurate the anomaly detection results are, the more effective the anomaly localization is. Therefore, the data with different noise levels is almost the same for RAPMiner. We need to ensure that the anomaly detection results are as accurate as possible, which also belongs to anomaly detection research. The effectiveness of RAPMiner and baseline methods on squeeze-B0 are shown in Fig. 8(a). RAPMiner, Squeeze, and FP-growth achieve comparable performance on squeeze-B0. Among them, RAPMiner shows the best $F1\text{-}score$ (i.e. $(1,1)$-1.0, $(1,2)$-0.995 and $(1,3)$-0.985) on the groups in which the RAPs are one-dimensional, and also achieves the best $F1\text{-}score$ on the groups of three-dimensional RAPs $(3,1)$ and $(3,2)$, which is 1.0 and 0.967 respectively. Squeeze achieves the best $F1\text{-}score$ of 0.970 and 0.982, respectively, on the groups of $(2,2)$ and $(2,3)$. The association rule mining based on FP-growth achieves the best $F1\text{-}score$ on $(2,1)$ and $(3,3)$ groups, which are 1.0 and 0.963 respectively. Adtributor only performs well on groups of one-dimensional RAPs, which is consistent with the assumption that its root anomaly patterns are located in one-dimensional cuboids. iDice achieves the inferior performance in all groups.

2) The Effectiveness on RAPMD Dataset: RAPMD does not group data by dimension or number of failures to avoid the desirable root anomaly patterns. We calculate $RC@3$, $RC@4$, and $RC@5$ of RAPMiner and Baseline Methods. Since the Squeeze algorithm can not return a specified number of results on RAPMD, we keep the same results for these three metrics of Squeeze. As shown in Fig. 8(b), RAPMiner achieves the best performance (above 80%) with regard to all of $RC@3$, $RC@4$ and $RC@5$ on RAPMD, which is at least 10% higher than the sub-optimal method, i.e., the association rule mining implemented with FP-growth. It can also be seen that the Squeeze method is less effective on RAPMD, due to the fact that it only works on datasets that follow the desirable assumptions. Adtributor only works on RAPMD datasets when the RAPs are one-dimensional, and its $RC@k$ can reach about 33%.

F. Efficiency Comparison

As shown in Fig. 9(a), Since Adtributor only searches one-dimensional cuboids and its search space is very small, its efficiency on groups $(1,1)$, $(1,2)$ and $(1,3)$ on the Squeeze-B0 dataset is the highest among all methods with the magnitude of running time in anomaly localization around 10^{-2}, but its effectiveness on groups $(1,2)$ and $(1,3)$ is too low. In contrast, the efficiency of the proposed RAPMiner can reach sub-optimal on the premise of ensuring the effectiveness of the algorithm on groups $(1,1)$, $(1,2)$ and $(1,3)$, and the magnitude of the running time to localize anomalies is 10^{-1}. In addition, it can also achieve optimal effectiveness and guarantee the efficiency simultaneously on groups $(2,1)$, $(2,2)$ and $(2,3)$, with the order of magnitude for time consumption ranging from 10^{-1} to 10^{0}. Squeeze can achieve the best efficiency in groups $(3,1)$, $(3,2)$ and $(3,3)$. iDice shows the worst efficiency among these algorithms and it takes more than 40s on average to complete the anomaly localization.It should be noted that the efficiency of RAPMiner is not related to the total number of attributes, but the number of attributes contained in the RAPs, because the redundant attributes can be deleted by Algorithm 1, and time consumption will increase with the increasing layer in top-down BFS search.

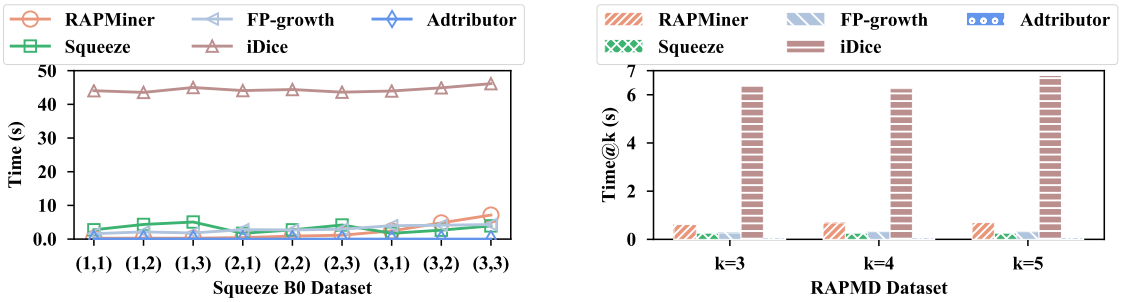
Fig. 9(b) shows, RAPMiner does not achieve the best efficiency on RAPMD, because this dataset simulates the actual anomaly localization scenario where there are many 3-dimensional RAPs in the failures, and the efficiency of RAPMiner in searching high-dimensional RAPs e.g., 3-dimensional, is inferior to 1-dimensional or 2-dimensional, so its efficiency on RAPMD is slightly lower than that of Squeeze and FP-growth. Despite this, it is still in an acceptable range, which is equivalent to sacrificing some time for the effective-



(a) $F1\text{-}score$ comparison on Dataset of Squeeze-B0

(b) $RC@k$ comparison over RAPMD dataset

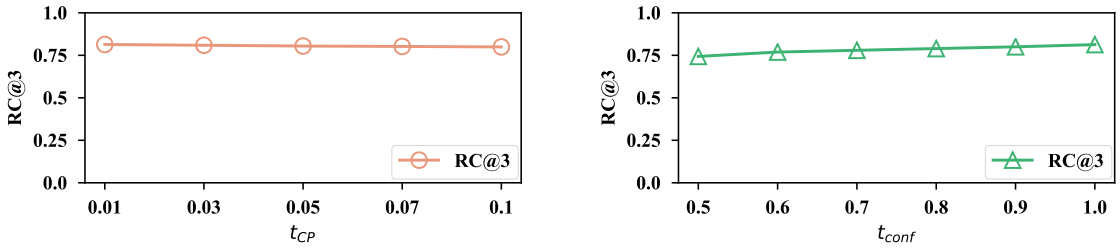
Fig. 8. Effectiveness comparison on Dataset of Squeeze-B0 and RAPMD



(a) Running time comparison on Squeeze-B0

(b) Running time comparison on RAPMD dataset

Fig. 9. Efficiency comparison on Dataset of Squeeze-B0 and RAPMD



(a) Sensitivity evaluation for parameter $\mathbf{t_{CP}}$

(b) Sensitivity evaluation for parameter $\mathbf{t_{conf}}$

Fig. 10. Sensitivity evaluation for parameters $\mathbf{t_{CP}}$ and $\mathbf{t_{conf}}$

TABLE VI
RESULTS OF THE EFFICIENCY IMPROVEMENT STUDY

Method	$RC@3(\%)$	Time(s)	Efficiency Improvement	Effectiveness Decreased
RAPMiner with Redundant Attribute Deletion	81.4	0.618	42.07%	4.87%
RAPMiner without Redundant Attribute Deletion	86.3	1.067		

ness of the algorithm. Although both Adtributor and Squeeze achieve higher efficiency, they show inferior effectiveness. And iDice shows low effectiveness and efficiency. The FP-growth-based association rule mining method is a sub-optimal method with high efficiency, but its effectiveness is about 10% lower than that of RAPMiner.

G. Parameter Sensitivity

We study the sensitivity of the threshold t_{CP} we employ in CP-based redundant attribute deletion to help prune redundant attributes and the confidence threshold t_{conf} we adopt in AC-guided layer-by-layer top-down search to judge whether the attribute combination is anomalous or not.

First, we set different values to parameter t_{CP} to evaluate its sensitivity. Since the classification power represents the ability to reduce the entropy on the dataset of the attribute, and our goal is to delete the attributes which are inferior in reducing the entropy, i.e., the redundant attributes or attributes unrelated to the root anomaly patterns. Hence, the threshold t_{CP} should not be too large. We usually set it to less than 0.1 (10%). Fig. 10(a) shows that with the increase of the threshold t_{CP}, the $RC@3$ slightly decreases, which means the performance of RAPMiner is not very sensitive to the threshold t_{CP}.

In addition, we also evaluate the sensitivity of the threshold t_{conf} by setting the different values. Since the attribute combination with higher anomaly confidence is more likely to

be anomalous, this threshold should not be too small. Thus several values greater than 0.5 (50%) are selected to illustrate the sensitivity. As shown in Fig. 10(b), with the increase of t_{conf}, the $RC@3$ also increases slightly, which states that the performance of RAPMiner is relatively stable w.r.t. t_{conf}.

The parameter sensitivity analysis demonstrates there is a large room for us to choose the threshold t_{CP} and t_{conf}.

H. The Efficiency Improvement of Redundant Attributes Deletion

To illustrate the efficiency improvement of the proposed CP-based redundant attribute deletion, we conduct two experiments on RAPMD. One is the RAPMiner with Redundant Attribute Deletion and the other is the RAPMiner without Redundant Attribute Deletion. As reported in Table VI, the redundant attribute deletion can improve the average efficiency of RAPMiner by 42.07%, while the effectiveness is only decreased by about 4.87%. The results verify the usefulness of the proposed CP-based redundant attribute deletion.

VI. Related Work

The research of anomaly localization mainly includes root cause mining based on multi-dimensional KPIs and root cause inference based on the dependency graph.

Root cause mining based on multi-dimensional KPI. Many existing studies have focused on root cause mining using multi-dimensional KPIs [13]–[15], [21], [22], [33]. Among them, Adtributor [13] mainly focuses on anomaly localization in the advertising system and assumes that the root anomaly patterns are only located in a one-dimensional cuboid, considering the characteristics of the advertising system. In this way, Adtributor only needs to traverse each attribute value of each attribute, and the size of the search space equals the number of attribute elements. Adtributor proposes three metrics to determine whether an attribute combination is the root cause or not: explanation power, succinctness, and surprise. In addition, Adtributor designs different localizing mechanisms for fundamental and derived KPIs. iDice [14] can conduct root cause mining in multi-dimensional attribute combinations. It narrows the search space by pruning off attribute combinations through the metric named "Impact", performs anomaly detection of attribute combinations via Change Detection, and conducts root cause mining using Isolation Power. However, iDice has inherent constraints and can not perform well in the case of many root anomaly patterns. HotSpot [21] can also conduct root cause mining in multi-dimensional attribute combinations. It designs a root cause metric named potential score and adopts the MCTS (Monte Carlo Tree Search) to mine the root causes of anomalies. However, it is only suitable for fundamental KPIs. Besides, HotSpot assumes that the root anomaly patterns are only located in one cuboid at a time, and descendant attribute combinations under the same root cause have the same anomaly magnitude. Squeeze [22] is a supplement and improvement to HotSpot. It proposes a potential generic score based on HotSpot's potential score, which can be applied to both fundamental and derived KPIs and allow multiple failures simultaneously. But the root anomaly patterns for each failure locate only in the same cuboid. In addition, Squeeze also assumed that attribute combinations under the same failure have the same anomaly magnitude, and different attribute combinations under various failures have different anomaly magnitude. However, the anomaly magnitude of attribute combinations may be the same under different failures and vice versa. [15] mentioned that association rule mining is used to search for the root cause. There are many ways to realize association rule mining, such as Apriori and FP-growth. The efficiency of different implementation methods varies greatly and the performance of some methods may be sensitive to hyper-parameters. FluxRank [33] may be clustering and ranking thousands or even millions of KPIs of different attribute combinations to infer the system's failures from the selected digests directly.

Although these multi-dimensional KPI-based anomaly localization methods perform well in specific scenarios, e.g., the advertising system, software application, etc., they are not effective in real-world CDN due to their desirable assumptions. Compared with the existing methods, ARPMiner is more practical in localizing root anomaly patterns effectively and efficiently without strict assumptions.

Root cause inference based on dependency graph. In this line of research, the fault localization research mainly focuses on constructing the dependency graph between system modules and the inference of the root cause of the fault based on the constructed dependency graph. Current relevant studies include but are not limited to [28]–[30], [34], [35], etc. Our work can be used as a supplement to the root cause inference based on the dependency graph because it can find the exact scope of failures and thus assist root cause analysis.

VII. Conclusion

In this paper, a novel root anomaly pattern Miner (RAPMiner) is proposed to tackle the anomaly localization challenges in real-world CDN scenario. RAPMiner considers the practical distribution features of anomaly patterns to avoid the desirable assumptions so that it is more practical. For the first time, RAPMiner combines a classification power-based redundant attribute deletion with anomaly confidence guided layer by layer top-down search to prune the non-root cause attribute combinations and avoid searching for anomaly but non-root patterns respectively, thus effective in narrowing the search space. The extensive experiments show the effectiveness and efficiency of RAPMiner compared with the state-of-the-art baselines, especially in the CDN scenario.

Acknowledgment

This work was supported in part by National Natural Science Foundation of China under Grant 61771469, and the Cooperation project between Chongqing Municipal undergraduate universities and institutes affiliated to CAS (HZ2021015). We thank our shepherd Prof. Shuai Hao, and the anonymous reviewers of DSN'22 for their insightful comments, which helped improve the paper.

REFERENCES

[1] L. Dai, T. Lin, C. Liu, B. Jiang, Y. Liu, Z. Xu, and Z.-L. Zhang, "SDFVAE: Static and dynamic factorized vae for anomaly detection of multivariate cdn kpis," in *WWW '21: The Web Conference*, 2021.

[2] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *ACM SIGKDD*, 2019, pp. 2828–2837.

[3] M. Sun, Y. Su, S. Zhang, Y. Cao, Y. Liu, D. Pei, W. Wu, Y. Zhang, X. Liu, and J. Tang, "Ctf: Anomaly detection in high-dimensional time series with coarse-to-fine model transfer," in *IEEE INFOCOM 2021*, 2021, pp. 1–10.

[4] W. Chen, H. Xu, Z. Li, D. Pei, J. Chen, H. Qiao, Y. Feng, and Z. Wang, "Unsupervised anomaly detection for intricate kpis via adversarial training of VAE," in *IEEE INFOCOM*, 2019, pp. 1891–1899.

[5] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *WWW '18*, 2018, pp. 187–196.

[6] Y. Su, Y. Zhao, M. Sun, S. Zhang, X. Wen, Y. Zhang, X. Liu, X. Liu, J. Tang, W. Wu, and D. Pei, "Detecting outlier machine instances through gaussian mixture variational autoencoder with one dimensional cnn," *IEEE Transactions on Computers*, pp. 1–1, 2021.

[7] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, and M. Feng, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *ACM IMC '15*, 2015, pp. 211–224.

[8] Y. Chen, R. Mahajan, B. Sridharan, and Z. Zhang, "A provider-side view of web search response time," in *ACM SIGCOMM '13*, 2013.

[9] H. Yan, A. Flavel, Z. Ge, A. Gerber, D. Massey, C. Papadopoulos, H. Shah, and J. Yates, "Argus: End-to-end service anomaly detection and localization from an isp's point of view," in *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*, A. G. Greenberg and K. Sohraby, Eds. IEEE, 2012, pp. 2756–2760.

[10] P. Bahl, R. Chandra, A. G. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang, "Towards highly reliable enterprise network services via inference of multi-level dependencies," in *ACM SIGCOMM '07*. ACM, 2007, pp. 13–24.

[11] M. Ma, S. Zhang, D. Pei, X. Huang, and H. Dai, "Robust and rapid adaption for concept drift in software system anomaly detection," in *29th IEEE International Symposium on Software Reliability Engineering, ISSRE 2018, Memphis, TN, USA, October 15-18, 2018*. IEEE Computer Society, 2018, pp. 13–24.

[12] S. Zhang, Y. Liu, D. Pei, Y. Chen, X. Qu, S. Tao, and Z. Zang, "Rapid and robust impact assessment of software changes in large internet-based services," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies, CoNEXT 2015, Heidelberg, Germany, December 1-4, 2015*, F. Huici and G. Bianchi, Eds. ACM, 2015, pp. 2:1–2:13.

[13] R. Bhagwan, R. Kumar, R. Ramjee, G. Varghese, S. Mohapatra, H. Manoharan, and P. Shah, "Adtributor: Revenue debugging in advertising systems," in *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2014, Seattle, WA, USA, April 2-4, 2014*. USENIX Association, 2014, pp. 43–55.

[14] Q. Lin, J. Lou, H. Zhang, and D. Zhang, "idice: problem identification for emerging issues," in *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, L. K. Dillon, W. Visser, and L. A. Williams, Eds. ACM, 2016, pp. 214–224.

[15] F. Ahmed, J. Erman, Z. Ge, A. X. Liu, J. Wang, and H. Yan, "Detecting and localizing end-to-end performance degradation for cellular data services based on TCP loss ratio and round trip time," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3709–3722, 2017.

[16] C. Lou, P. Huang, and S. Smith, "Understanding, detecting and localizing partial failures in large system software," in *17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020*, R. Bhagwan and G. Porter, Eds. USENIX Association, 2020, pp. 559–574.

[17] S. Burnett, L. Chen, D. A. Creager, M. Efimov, I. Grigorik, B. Jones, H. V. Madhyastha, P. Papageorge, B. Rogan, C. Stahl, and J. Tuttle, "Network error logging: Client-side measurement of end-to-end web service reliability," in *17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020*, R. Bhagwan and G. Porter, Eds. USENIX Association, 2020, pp. 985–998.

[18] J. Dilley, B. M. Maggs, J. Parikh, H. Prokop, R. K. Sitaraman, and W. E. Weihl, "Globally distributed content delivery," *IEEE Internet Comput.*, vol. 6, no. 5, pp. 50–58, 2002.

[19] J. Jiang, S. Sun, V. Sekar, and H. Zhang, "Pytheas: Enabling data-driven quality of experience optimization using group-based exploration-exploitation," in *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, A. Akella and J. Howell, Eds. USENIX Association, 2017, pp. 393–406.

[20] J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, and H. Zhang, "CFA: A practical prediction system for video qoe optimization," in *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*, K. J. Argyraki and R. Isaacs, Eds. USENIX Association, 2016, pp. 137–150.

[21] Y. Sun, Y. Zhao, Y. Su, D. Liu, X. Nie, Y. Meng, S. Cheng, D. Pei, S. Zhang, X. Qu, and X. Guo, "Hotspot: Anomaly localization for additive kpis with multi-dimensional attributes," *IEEE Access*, vol. 6, pp. 10 909–10 923, 2018.

[22] Z. Li, D. Pei, C. Luo, Y. Zhao, Y. Sun, K. Sui, X. Wang, D. Liu, X. Jin, and Q. Wang, "Generic and robust localization of multi-dimensional root causes," in *30th IEEE International Symposium on Software Reliability Engineering, ISSRE 2019, Berlin, Germany, October 28-31, 2019*, K. Wolter, I. Schieferdecker, B. Gallina, M. Cukier, R. Natella, N. R. Ivaki, and N. Laranjeiro, Eds. IEEE, 2019, pp. 47–57.

[23] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," in *ICML 2016 Anomaly Detection Workshop*, 2016.

[24] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Söderström, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *ACM SIGKDD '18*, 2018, pp. 387–395.

[25] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.

[26] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[27] I. Kononenko and S. J. Hong, "Attribute selection for modelling," *Future Gener. Comput. Syst.*, vol. 13, no. 2-3, pp. 181–195, 1997.

[28] Y. Pan, M. Ma, X. Jiang, and P. Wang, "Faster, deeper, easier: crowdsourcing diagnosis of microservice kernel failure from user space," in *ISSTA '21: 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, Denmark, July 11-17, 2021*, C. Cadar and X. Zhang, Eds. ACM, 2021, pp. 646–657.

[29] Y. Meng, S. Zhang, Y. Sun, R. Zhang, Z. Hu, Y. Zhang, C. Jia, Z. Wang, and D. Pei, "Localizing failure root causes in a microservice through causality inference," in *28th IEEE/ACM International Symposium on Quality of Service, IWQoS 2020, Hangzhou, China, June 15-17, 2020*. IEEE, 2020, pp. 1–10.

[30] D. Liu, C. He, X. Peng, F. Lin, C. Zhang, S. Gong, Z. Li, J. Ou, and Z. Wu, "Microhecl: High-efficient root cause localization in large-scale microservice systems," in *43rd IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP) 2021, Madrid, Spain, May 25-28, 2021*. IEEE, 2021, pp. 338–347.

[31] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Morgan Kaufmann, 1994, pp. 487–499.

[32] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *SIGMOD Rec.*, vol. 29, no. 2, p. 1–12, may 2000.

[33] P. Liu, Y. Chen, X. Nie, J. Zhu, S. Zhang, K. Sui, M. Zhang, and D. Pei, "Fluxrank: A widely-deployable framework to automatically localizing root cause machines for software service failure mitigation," in *30th IEEE International Symposium on Software Reliability Engineering, ISSRE 2019, Berlin, Germany, October 28-31, 2019*, K. Wolter, I. Schieferdecker, B. Gallina, M. Cukier, R. Natella, N. R. Ivaki, and N. Laranjeiro, Eds. IEEE, 2019, pp. 35–46.

[34] P. Chen, Y. Qi, P. Zheng, and D. Hou, "Causeinfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems," in *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*. IEEE, 2014, pp. 1887–1895.

[35] M. Ma, Z. Yin, S. Zhang, S. Wang, C. Zheng, X. Jiang, H. Hu, C. Luo, Y. Li, N. Qiu, F. Li, C. Chen, and D. Pei, "Diagnosing root causes of intermittent slow queries in large-scale cloud databases," *Proc. VLDB Endow.*, vol. 13, no. 8, pp. 1176–1189, 2020.