Generic and Robust Localization of Multi-Dimensional Root Causes

Zeyan Li[†‖], Chengyang Luo[†], Yiwei Zhao[†], Yongqian Sun[‡*], Kaixin Sui[§]
Xiping Wang[†], Dapeng Liu[§], Xing Jin[¶], Qi Wang[¶], Dan Pei[†‖]
[†]Tsinghua University, [‡]Nankai University, [§]BizSeer, [¶]China Construction Bank
[‖]Beijing National Research Center for Information Science and Technology (BNRist)

***Abstract*—Operators of online software services periodically collect various measures with many attributes. When a measure becomes abnormal, indicating service problems such as reliability degrade, operators would like to rapidly and accurately localize the root cause attribute combinations within a huge multi-dimensional search space. Unfortunately, previous approaches are not *generic* or *robust* in that they all suffer from impractical root cause assumptions, handling only directly collected measures but not derived ones, handling only anomalies with significant magnitudes but not those insignificant but important ones, requiring manual parameter fine-tuning, or being too slow.**

This paper proposes a generic and robust multi-dimensional root cause localization approach, *Squeeze*, that overcomes all above limitations, the first in the literature. Through our novel "bottom-up then top-down" searching strategy and the techniques based on our proposed generalized ripple effect and generalized potential score, *Squeeze* is able to reach a good trade-off between search speed and accuracy in a generic and robust manner. Case studies in several banks and an Internet company show that *Squeeze* can localize root causes much more rapidly and accurately than the traditional manual analysis. Furthermore, our extensive experiments on semi-synthetic datasets show that the F1-score of *Squeeze* outperforms previous approaches by 0.4 on average, while its localization time is only about 10 seconds.

***Index Terms*—multi-dimensional, root cause localization, generalized ripple effect, generalized potential score**

I. Introduction

Operators of online software services (*e.g.*, online shopping and Internet company) periodically (*e.g.*, every minute) collect various measures to monitor and troubleshoot service's performance and reliability problems [1], [2]. Some measures are directly collected from raw logs (*e.g.*, total dollar mount of orders in Table I), and are called fundamental measures [3]. Other measures derived from fundamental measures by some functions (*e.g.*, the average dollar amount per order in Table II) are called derived measures [3]. Measures are usually collected with many attributes (*e.g.*, Province, ISP) whose values are categorical (*e.g.*, Beijing or Shanghai for attribute ISP). The measure can be calculated for each attribute combination (*i.e.*, a conjunction of attribute and attribute value pairs). *e.g.*, in the third row of Table I, the measure for attribute combination (Province=Shanghai$\wedge$ISP=China Unicom) is 30.

When a measure's real value deviates from its expected (forecast) value, this measure encounters anomalies (see the

Yongqian Sun[*] is the corresponding author.

TABLE I: Fundamental measure example: total dollar amount of orders. The bold lines are abnormal. The root cause is (Province=Beijing).

Province	ISP	real value	forecast value
Beijing	**China Mobile**	**5**	**10**
Beijing	**China Unicom**	**10**	**20**
Shanghai	China Unicom	30	31
Guangdong	China Mobile	10	9.8
Zhejiang	China Unicom	2	2
Guangdong	China Unicom	200	210
Shanxi	China Unicom	20	22
Jiangsu	China Unicom	200	203
Tianjin	China Mobile	41	43
Total		518	550.8

TABLE II: Derived measure example: the average dollar amount per order ($\frac{\text{total dollar amount of orders}}{\text{the number of orders}}$). The bold lines are abnormal. The root cause is (Province=Beijing).

Province	ISP	real value	forecast value
Beijing	**China Mobile**	**50/20**	**10/10**
Beijing	**China Unicom**	**120/60**	**24/30**
Shanghai	China Unicom	30/30	31/30
Guangdong	China Mobile	10/21	9.8/20
Zhejiang	China Unicom	2/2	2/2
Total		212/133	76.8/92

bold lines in Table I and Table II). In such case, operators would like to rapidly and accurately localize the *root cause* attribute combinations, *e.g.*, (Province=Beijing) for both Table I and Table II. Localization of multi-dimensional root causes is thus critical to troubleshooting and mitigating the software service performance and reliability anomalies.

The fundamental challenge of multi-dimensional root cause localization is its huge search space due to the large number of attribute combinations and that in theory the root cause can be a set of any number of attribute combinations. Thus, even for the toy examples in Table I and Table II, there exists $2^{7+2+9}-1$ and $2^{4+2+5}-1$ potential root causes respectively. Suppose there are d attributes and each attribute has l distinct values, then there can be $\sum_{i=1}^{d}\binom{d}{i}l^i = (l+1)^d - 1$ valid attribute combinations and the number of potential root cause can be $2^{(l+1)^d-1}-1$. According to the experiences of operators that we worked with, typically d is less than or about ten, l is several tens. When $d=5$ and $l=10$, then there will be $2^{161050}-1$ potential root causes.

Previous multi-dimensional root cause localization approaches [3]–[7] proposed various techniques to reduce the

TABLE III: Qualitative comparison of related works. Detailed reviews are in Section VII.

Algorithms	**Root Cause Assumption**	**Measure**	**Change Magnitude**	**Parameter Fine Tuning**	**Time Cost**	**Method**
Adtributor [3]	single attribute (in one first-layer cuboid)	**fundamental & derived (quotient)**	significant	**no**	**very short**	top-down
R-Adtributor [4]	**none**	**fundamental & derived (quotient)**	significant	yes	**short**	top-down
iDice [5]	one or two attribute combinations	fundamental only	significant	**no**	**very short**	top-down
Apriori [6]	**none**	**fundamental & derived**	**any**	yes	always too long	bottom-up
HotSpot [7]	all attribute combinations of the root cause in one cuboid	fundamental only	significant	**no**	sometimes long	top-down
Squeeze	**those which cause the same changes are in one cuboid**	**fundamental & derived (quotient, product)**	**any**	**no**	**short**	bottom-up then top-down

search space, but they all suffer some limitations (summarized in Table III) and are not generic or robust. Adtributor [3] and iDice [5] have assumptions on root causes that are too strong and impractical. HotSpot [7] and iDice can only handle fundamental measures, and Adtributor and R-Adtributor [4]'s approaches for derived measures are very different from the ones for fundamental measures. iDice, HotSpot, Adtributor and R-Adtributor all ignore attribute combinations whose anomaly magnitudes are insignificant. (*e.g.*, in Table I, where the abnormal lines change by 15, which is insignificant relative to the total amount 518). However, such anomalies are important in some cases (*e.g.*, bank transactions failure of a small number of users). Apriori and R-Adtributor stop searching with arbitrary and hard-to-tune conditions with no clear physical meanings in the context of root cause localization (*e.g.*, the minimum support threshold), thus require manual parameter fine-tuning for different services or the same service over time, thus are impractical. Apriori [6] is too time-consuming.

Our **design goals** for multi-dimensional root cause localization are *generic* and *robust*: 1) consistently handle fundamental and derived measures; 2) handle both significant and insignificant anomaly magnitudes; 3) do not require parameter fine-tuning; 4) be consistently fast, and 5) have no impractical root cause assumptions. The core ideas and major contributions of our proposed approach, *Squeeze*, are the following.

First, for the first time in the literature, we propose *generalized ripple effect* (GRE) for both fundamental and derived measures, which captures magnitude relationship between the root cause attribute combinations and its "descendant" attribute combinations. GRE has two major differences from the the *ripple effect* observed by [7]: 1) ripple effect was considered by [7] to hold *only for fundamental measures*. We mathematically prove that GRE holds for a derived measure as long as the ripple effect holds for its underlying fundamental measures 2) the ripple effect in [7] does not work for zero forecast values (not uncommon in practice), but we make improvements so that GRE works for zero forecast values. Furthermore, we show that GRE holds true in the real world through case studies on an Internet company and several large banks. **GRE and its confirmation in reality are our first contribution**.

Second, we attribute previous approaches' limitations to the fact that, to reduce the search space, they all use *either* top-down method [3]–[5], [7] *or* bottom-up method [6], but not both. In contrast, *Squeeze* first searches bottom-up to narrow down the search space, within which *Squeeze* then does top-down search to localize the root cause attribute combinations. In the bottom-up step, *Squeeze* filters out most normal attribute combinations and groups potential abnormal attribute combinations into clusters according to GRE. In the top-down step, *Squeeze* uses a heuristic method (with only one robust parameter) to efficiently search for the root causes within those clusters output by the bottom-up step. **Our novel "bottom-up&top-down" method (hence the name *Squeeze*) enables *Squeeze* to become the first generic and robust approach in the literature for multi-dimensional root cause localization, which is our second contribution**.

We extensively evaluate *Squeeze* by comparing with previous works on several semi-synthetic datasets. The experiments show that the F1-score of *Squeeze* significantly outperforms the others by about **0.4**. *Squeeze* is also efficient and consistently costs only about 10 seconds in all cases. Case studies in several banks and an Internet company show that *Squeeze* can localize root causes much more rapidly and accurately than traditional manual analysis. **This is our third contribution**.

II. Background

TABLE IV: Raw Transaction Log Example for Table I

Order ID	Timestamp	Dollar Amount	Province	ISP

Table V lists the the definitions, notations, and examples of the important terms used in the paper. *Measure* is the value of interest, *e.g.*, total dollar amount of orders in Table I, average dollar amount per order in Table II, and success rate in the case studies in Section V-A and Section V-B. Measures are calculated periodically with many *attributes* (*e.g.*, ISP) based on raw logs of transactions (*e.g.*, Table IV). Each attribute will have several discrete potential values (*e.g.*, ChinaUnicom, ChinaMobile for attribute ISP). A conjunction of several (attribute, attribute value) pairs where an attribute can appear in at most one pair is called *attribute combination*.

A *leaf attribute combination* (also referred to as *leaf* for simplicity) is an attribute combination which includes all attributes. From raw logs like Table IV, given a time window, we sum the dollar amount of all transactions with the same leaf attribute combination to obtain its measure. *e.g.*, each line except for the last one in Table I corresponds to a leaf attribute combination and its dollar amount measure

TABLE V: Summary of Terms

Term	Definition	Notation	Example
Measure	The measure of interest	M	total dollar amount of orders, average dollar amount per order
Fundamental Measure	Additive measures	-	total dollar amount of orders, the number of orders, the number of successful transactions, total search response time (SRT)
Derived Measure	Measure derived from fundamental measures	-	average dollar amount per order ($\frac{\text{total dollar amount of orders}}{\text{the number of orders}}$), success rate ($\frac{\text{\#success transactions}}{\text{\#total transactions}}$), average SRT ($\frac{\text{total SRT}}{\text{\#page views}}$)
Attribute	All categories	A	Province, ISP in Table I
Attribute Value	Potential values for each attribute	a	Beijing, Shanghai, Guangdong for attribute Province
Attribute Combination	A conjunction of (attribute, attribute value) pairs	e	(Province=Beijing) and (Province=Shanghai$\wedge$ISP=China Mobile) are both valid. (Province=Shanghai$\wedge$Province=Beijing) is not valid.
Leaf Attribute Combination	An attribute combination such that all potential attributes are specified a value	e	(Province = Beijing $\wedge$ ISP = China Unicom), (Province=Beijing$\wedge$ISP=China Mobile) in Table I
Cuboid	Set of attribute combinations whose corresponding attribute combinations have the same keys	C	$C_{\text{ISP}} = \{$(ISP = China Mobile), (ISP = China Unicom)$\}$, $C_{\text{Province,ISP}}$=$\{$(Province=Beijing,ISP=China Mobile), (Province=Beijing,ISP=China Unicom), (Province=Shanghai,China Unicom), (Province=Guangdong, ISP=China Mobile)...$\}$
Descent	$e_1$is descended from e_2 means the attribute combination of e_1 contains that of e_2	-	(Province = Beijing $\wedge$ ISP = China Unicom) is descended from (Province = Beijing), but is not descended from (Province=Shanghai)
Real Value	The measure values that are collected from real world.	$v(\cdot)$	In Table I, v(Province = Beijing $\wedge$ ISP = China Unicom) = 10,
Forecast Value	Measures' expected values.	$f(\cdot)$	In Table I, f(Province = Beijing $\wedge$ ISP = China Unicom) = 20

for the given time window. Such additive measures, which are collected directly from raw logs, are called *fundamental measures*. They can be sliced along different attributes. For example, in Table I, the dollar amount of attribute combination (Province=Beijing) (not shown in the table) is the sum over of those of (Province=Beijing$\wedge$ISP=China Mobile) and (Province=Beijing$\wedge$ISP=China Unicom), while the total amount (last line in Table I) is the sum over those of all leaves. There are also *derived measures* which are functions of fundamental measures. They are typically non-additive. For example, from raw logs like Table IV, given a time window, we count the number of transactions (orders) for each leaf to obtain the number of orders for the given time window. Then the average dollar amount per order in Table II is the total dollar amount of orders divided by the number of orders.

All the attribute combinations with the same set of attributes (regardless of the attribute values) form a set of attribute combinations called *cuboid* [8] (see 4th last row in Table V for examples). If there are d attributes in total, there would be $2^d - 1$ cuboids. Cuboids can form a multi-layer graph, as Fig. 1 shows. A cuboid's layer is the number of attributes it uses. If there are $\hat{d}$ attributes in a cuboid and each has l distinct values, there would be $l^{\hat{d}}$ attribute combinations in it. We say an attribute combination e_1 is *descent* from e_2 if e_1's conjunction contains all those of e_2 (see 3rd last row in Table V for examples).

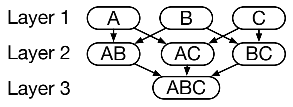

Fig. 1: A cuboid graph with 3 attributes: A, B, C.

The *real value* of an attribute combination is the measure values directly collected/calculated based on the raw transaction logs, and the *forecast value* is its expected value based on some forecasting algorithms. The last two columns in Table I and Table II are real and forecast values, respectively. An attribute combination is considered *abnormal* for a given time window if its real value deviates *significantly enough* from its expected value. Note that although value forecasting interacts with root cause localization, it is not in our studied scope of root cause localization. There are lots of time series forecasting algorithms that handles value forecasting, from simple statistical algorithms [9]–[12] to advanced algorithms [13]–[15], or forecasting for multi-dimensional attributes [6]. The choice of appropriate forecasting algorithm sometimes needs be based on the specific data [16]. To prevent fine-tuning forecasting algorithm from hurting the robustness of localization, in this paper, we simply use MA (moving average) for all scenarios. Our experiments in Section VI-E show that our proposed localization approach is robust against MA's various forecasting errors on different datasets.

Real values and forecast values are denoted by $v(\cdot)$ and $f(\cdot)$ respectively. Without loss of generality, we assume both v and f are non-negative, since in practice measures are non-negative. We only extract real and forecast values for all leaves from raw logs directly. For a fundamental measure, a non-leaf attribute combination's real (forecast) value is the sum over all leaves that descended from this non-leaf attribute combination. In other words, $v(e) = \sum_{e' \in \text{descent}(e)} v(e'), f(e) = \sum_{e' \in \text{descent}(e)} f(e')$. For a derived measure, a non-leaf attribute combination's real (forecast) value is the function of its corresponding fundamental measures' values for this attribute combination, *i.e.*, if $v{=}h(v_1, ..., v_n)$, then $v(e){=}h(v_1(e), ..., v_n(e))$, where v is a derived measure from fundamental measures $\{v_1, v_2, ..., v_n\}$ and $h(\cdot)$ is the function that produces v. For convenience, we slightly extend the definition of operator v and f. If S is a set of attribute combinations which are in the same cuboid, and the measure is a fundamental measure, $v(S){=}\sum_{e \in S} v(e)$ and $f(S){=}\sum_{e \in S} f(e)$. If it is a derived measure, $v(S){=}h(v_1(S), ..., v_n(S))$ and $f(S){=}h(f_1(S), ..., f_n(S))$.

III. Generalized Ripple Effect

We propose *generalized ripple effect* (GRE), which captures the relationship of attribute combinations' abnormal magnitudes caused by the same root cause. GRE holds for both fundamental and derived measures and can handle zero forecast values. GRE is a key enabler of *Squeeze*'s genericness and robustness, and in Section V we will show that it holds true in the real world through industrial case studies.

A. Background of Ripple Effect

Ripple effect, first observed by [7] for **fundamental** measures only, captures the relationship of attribute combinations' abnormal magnitudes caused by the same root cause. The intuition of ripple effect is that all attribute combinations affected by the same root cause will change by the same proportion. It says that, given a root cause S, which is a subset of cuboid C, the abnormal magnitude of any affected attribute combination e (in S or descended from any attribute combination in S) satisfies the following equation:

$$\frac{f(e)-v(e)}{f(e)}=\frac{f(S)-v(S)}{f(S)}, \tag{1}$$

where $f(S)$ ($v(S)$) denotes the total forecast (real) value of all **leaf** attribute combinations in S. For example, in Table I, the root cause is S={(Province=Beijing)} in cuboid C_{province}. Therefore, if e_1=(Province=Beijing∧ISP=China Unicom), then $\frac{f(S)-v(S)}{f(S)}=\frac{\sum_{e'\in S}(f(e')-v(e'))}{\sum_{e'\in S}f(e')}=0.5=\frac{f(e_1)-v(e_1)}{f(e_1)}$

B. Generalizing Ripple Effect for Derived Measures

Ripple effect was considered by [7] to hold *for only fundamental measures*. In this section, we mathematically prove that GRE holds for a derived measure as long as the ripple effect holds for its underlying fundamental measures. Since most common derived measures are the quotient of two fundamental measures, without much loss of generality, we provide proof of GRE for such derived measures.

Consider three measures, M_1, M_2, M_3, where M_1 and M_2 are fundamental measures and $M_3=\frac{M_1}{M_2}$. Since M_1 and M_2 are fundamental measures, M_1 and M_2 satisfies ripple effect (1). It can be proved the M_3 also satisfies ripple effect: Note that the real value and forecast value of a derived measures are defined by its component fundamental measures: $f_{M_3}(S)=\frac{f_{M_1}(S)}{f_{M_2}(S)}\neq\sum_{e'\in S}f_{M_3}(e')$. Therefore,

$$\Delta_{M_3}(e)=\frac{f_{M_1}(e)}{f_{M_2}(e)}-\frac{v_{M_1}(e)}{v_{M_2}(e)}=\frac{\Delta_{M_1}(e)v_{M_2}(e)-\Delta_{M_2}(e)v_{M_1}(e)}{v_{M_2}(e)f_{M_2}(e)}$$

$$\text{similarly,}\quad \Delta_{M_3}(S)=\frac{\Delta_{M_1}(S)v_{M_2}(S)-\Delta_{M_2}(S)v_{M_1}(S)}{v_{M_2}(S)f_{M_2}(S)}$$

$$\because\text{ripple effect,}\quad \therefore\Delta_{M_i}(e)=\frac{f_{M_i}(e)}{f_{M_i}(S)}\Delta_{M_i}(S), i=1,2$$

$$\therefore\Delta_{M_3}\frac{f_{M_3}(e)}{f_{M_3}(S)}=\frac{\Delta_{M_1}(S)v_{M_2}(S)-\Delta_{M_2}(S)v_{M_1}(S)}{v_{M_2}(S)f_{M_2}(S)}\frac{f_{M_2}(S)}{f_{M_1}(S)}\frac{f_{M_1}(e)}{f_{M_2}(e)}$$

$$=\frac{\Delta_{M_1}(e)-\frac{\Delta_{M_2}(S)}{v_{M_2}(S)}v_{M_1}(e)}{f_{M_2}(e)}=\frac{\Delta_{M_1}(e)-\frac{\Delta_{M_2}(e)}{v_{M_2}(e)}v_{M_1}(e)}{f_{M_2}(e)}=\Delta_{M_3}(e)\quad\square$$

Therefore, for each root cause, there is also a constant k such that for any attribute combination under affection $\frac{v_{M_3}(e)}{f_{M_3}(e)}=k$. We call it as *generalized ripple effect* (GRE). Take Table II as an example, the true root cause is $S=\{(Province{=}Beijing)\}$. Then for $e_1=(Province{=}Beijing\wedge ISP{=}Mobile)$, $\frac{v_{M_3}(e_1)}{f_{M_3}(e_1)}=\frac{50}{20}/\frac{10}{10}=\frac{v_{M_3}(S)}{f_{M_3}(S)}=\frac{50+120}{20+60}/\frac{10+24}{10+30}=\frac{5}{2}$.

GRE also holds for products ($M_3{=}M_1{\cdot}M_2$), which can be proved similarly. We prove GRE for quotients and productions because they are the most common cases. The core idea of the our proof is *finite difference* [17]. A similar method can be applied when dealing with other types of derived measures.

Our results of GRE can cover fundamental measures (*e.g.*, total dollar amount of orders, page views, issue counts, total traffic) and most common derived measures (*e.g.*, average latency, average dollar amount per order, success rate).

C. Generalizing Ripple Effect for Zero Forecast Values

The ripple effect in [7] does not work for zero forecast values (*i.e.*, $f(S)=0$. To avoid this problem, we replace f with $\frac{f+v}{2}$. That is to say, GRE's formulation becomes

$$\frac{f(e)-v(e)}{f(e)+v(e)}=\frac{f(S)-v(S)}{f(S)+v(S)} \tag{2}$$

where the symbols have the same meaning as (1). If (1) holds, then it is obvious that (2) holds as well; if $f(S)=0$, then any affected attribute combination e should have infinitely large abnormal magnitudes or $f(e)=0$.

IV. *Squeeze* Approach

We propose an approach, *Squeeze*, for multi-dimensional root cause localization, whose design goals are listed in Section I and scope/architecture shown in Fig. 2.

A. Scope of this Paper

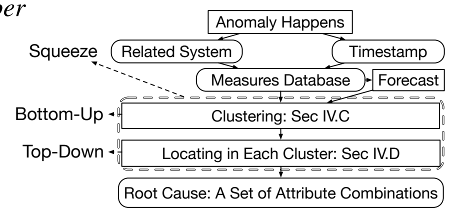

Fig. 2: *Squeeze*'s scope and architecture.

The double dashed box in Fig. 2 highlights the scope of *Squeeze*, which is **called on demand** when some anomaly (*e.g.*, success rate drops significantly) happens to a service at a particular time window, indicated by some calls, tickets, or alerts. Then operators query the measure's real values for leaf attribute combinations from the measure database, and call a forecasting algorithm (*i.e.*, Moving Average) **on demand** to obtain the measure's forecast values for all leaves for the given time window. In other words, *Squeeze* is called on demand with the *input* of a measure's real and forecast values of all leaves.

The output of *Squeeze* is a set of attribute combinations that explain all anomalies and are as succinct as possible (Occam's razor principle). For example, in Table I, both (Province=Beijing) and {(Province=Beijing∧ISP=China Mobile), (Province=Beijing∧ISP=China Unicom)} explain the anomaly, but we prefer (Province=Beijing) as the root case since it is more succinct.

The selection of the time series forecasting algorithm is out of the scope of *Squeeze* as mentioned in Section II. Similar to other multi-dimensional root cause localization approaches [3], [5], [7], casual inference is also out of scope.

B. Core Ideas

Different from all previous works [3]–[7], *Squeeze* employs a novel "bottom-up then top-down" searching strategy to achieve a good trade-off between speed and accuracy in a generic and robust manner. *Squeeze* first searches bottom-up to narrow down the search space, within which *Squeeze* then does top-down search to localize the root cause. In the bottom-up step, *Squeeze* filters out most normal attribute combinations and groups potential abnormal attribute combinations into clusters according to GRE. In the top-down step, *Squeeze* uses a heuristic method based on our proposed generalized potential score (GPS) to efficiently search for the root causes within those clusters output by the bottom-up step.

C. Bottom-Up Searching through Clustering

Squeeze first searches bottom-up to narrow down the search space. More specifically, *Squeeze* filters out most normal attribute combinations and then groups the remaining potentially abnormal ones into clusters according to GRE.

1) Deviation Based Filtering: Given the large number of leaf attribute combinations, when a service anomaly occurs, usually the number of "abnormal" leaf attribute combinations (with large deviations) is much less than those "normal" ones (with little deviations). The goal of this step is to filter out those normal leaf attribute combinations in order to reduce the number of leaf attribute combinations for the next step (clustering). One key challenge is how to automatically determine whether a leaf attribute combination is abnormal or not. Similar to those anomaly detection algorithms [13], [16], we use the forecast residuals (or *deviation*) to indicate the extent of changes, and apply a threshold to decide whether the change is an anomaly or not. Fig. 3 shows the cumulative distribution of leaf attribute combinations' deviations in an example service anomaly. The skewed distribution naturally allows us to apply the *knee-point* method [18] to *automatically* pick the knee-point as the anomaly threshold, on demand for each service anomaly. For example, the vertical dashed line in Fig. 3 marks the threshold given by knee method for this example.

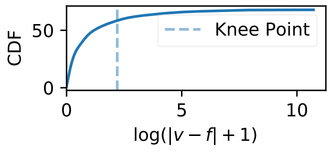

Fig. 3: Cumulative distribution of leaf attribute combinations' deviations, and knee point threshold selection.

2) Deviation Score Based Clustering: The deviation based filtering outputs a list of potentially abnormal leaf attribute combinations which *Squeeze* groups into clusters, each of which is a potential root cause.

Deviation score is defined as $d(e){:=}2\frac{f(e)-v(e)}{f(e)+v(e)}$. All leaves of the same root cause will have similar deviation scores according to GRE. Therefore, we can find which (potentially abnormal) leaf attribute combinations are affected by the same root cause by grouping leaf attribute combinations with similar deviation scores into the same cluster.

We use a simple density based clustering algorithm, as shown in Algorithm 1. First we get the histogram of the deviation scores. Then the relative maximums of the histogram are considered as the cluster centroids, and the closest relative minimums are considered as the cluster boundaries.

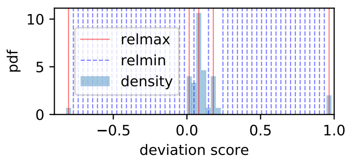

Fig. 4: Illustration of Algorithm 1 by *Case II* in Section V-A

Algorithm 1 Deviation Score Based Clustering by Density

```
1: procedure DensityCluster($arr$)
2:     $bins, hists \leftarrow$ histogram($arr$)
3:     $centers \leftarrow$ argrelmax($hists$)
4:     $boundaries \leftarrow$ argrelmin($hists$)
5:     $clusters \leftarrow []$
6:     for $center$ in $centers$ do
7:         $l \leftarrow$ last attribute combination in $boundaries$ s.t. $bins[l] < bins[center]$
8:         $r \leftarrow$ first attribute combination in $boundaries$ s.t. $bins[r] > bins[center]$
9:         $clusters \leftarrow clusters + \{x \in arr | l \le x \le r\}$
       return clusters
```

Algorithm 2 Localization in Cluster

```
1: procedure InClusterLocalization($cluster$)
2:     $root\_causes \leftarrow []$
3:     for $cuboid$ in all cuboids from top to bottom do
4:         $n\_ele \leftarrow$ the number of leaf attribute combinations descended from each attribute combination of $cuboid$ in $cluster$
5:         $n\_descents \leftarrow$ the number of all leaf attribute combinations descended from each attribute combination of $cuboid$
6:         sort attribute combinations in $cuboid$ by $\frac{n\_ele}{n\_descents}$ in descending order
7:         for $split$ in all valid splits do
8:             $score[split] \leftarrow$ GPS($split$)
9:         $index \leftarrow \text{argmax}_{split} score$
10:        $root\_cause \leftarrow attribute\_combinations[index]$
11:        $root\_causes \leftarrow root\_causes + [root\_cause]$
12:        if $root\_cause$'s $score \ge \delta$ then
13:            Stop search next layer
14:    sort $root\_causes$ with $score * C - n\_ele * cuboid\_layer$ in descending order
15:    return $root\_causes[0]$
```

D. Top-Down Localization within Each Cluster

The output of the bottom-up search is a list of leaf attribute combination clusters, each of which is a potential root cause with a different range of deviation scores. This output is the input of the top-down localization. Within each cluster, we use a top-down heuristic algorithm (shown in Algorithm 2) to efficiently search for the root cause attribute combinations.

1) Root Cause Assumption: Squeeze does have one assumption about root causes: we assume that the root cause attribute combination is always a subset of only one cuboid in each cluster outputted by Section IV-C. However, unlike the impractical root cause assumptions in [3] and [5], our assumption above about root causes is practical because: 1) one physical root cause brings the same abnormal magnitudes according to GRE, and it is almost impossible in practice there are several physical root causes which cause the same abnormal magnitudes at the same time; 2) in practice it is very rare that root cause attribute combinations which cause the same abnormal magnitudes require more than one cuboid.

2) Top-Down Localization Heuristics: In every cuboid, there is a set of attribute combinations that can cover all abnormal leaf attribute combinations in the cluster, and there is a subset of the **leaf** cuboid that can exactly match all abnormal leaf attribute combinations. Therefore, determining the right cuboid is important for localizing the root cause. The number of attributes is denoted by d, then there will be only $2^d - 1$ cuboids. Since d is usually less than 10 or about 10, so it is acceptable to enumerate all cuboids if it does not take too long to find the best solution in each cuboid.

Squeeze uses a heuristic method to find best subset of each cuboid. The key idea is that, if an attribute combination e in the current cuboid belongs to the root cause, all of its descent leaf attribute combinations should be abnormal and in the same cluster because of GRE. In other words, a necessary condition for an attribute combination belonging to the root cause is that most of its descent leaf attribute combinations are in the cluster. We denote the ratio of descent leaf attribute combinations in the cluster by *descent score*. We sort the attribute combinations of the cuboid by descent score in descending order, then the root cause should be the left part of the sorted attribute combinations with appropriate partition. For example, in the cuboid of attribute province of Table I, $(Province = Beijing)$'s descent score is 1 and those of all other attribute combinations are 0.

Now we only need to compare all potential partitions. Given a partition, the part with larger descent score are classified as abnormal and the other one as normal. A good partition should make the abnormal part contain most anomalies and the normal part contain few. Then, the abnormal leaf attribute combinations should follow GRE. Actually, according to GRE, the real value of an abnormal leaf attribute combination e under root cause S is expected to be $a_S(e){=}f(e){-}\frac{f(e)}{f(S)}(f(S){-}v(S)) = f(e)\frac{v(S)}{f(S)}$. Thus, the difference between the expected value and real value $|v(e) - a_S(e)|$ can be used to evaluate how well the root cause S follows GRE. Therefore, we propose *generalized potential score* (GPS), defined as follows:

$$GPS = 1 - \frac{\text{avg}(|\boldsymbol{v}(S_1) - \boldsymbol{a}(S_1)|) + \text{avg}(|\boldsymbol{v}(S_2) - \boldsymbol{f}(S_2)|)}{\text{avg}(|\boldsymbol{v}(S_1) - \boldsymbol{f}(S_1)|) + \text{avg}(|\boldsymbol{v}(S_2) - \boldsymbol{f}(S_2)|)} \quad (3)$$

where S_1 is the set of abnormal (*i.e.*, descent of the root cause) leaf attribute combinations of this cluster; S_2 is the set of all normal leaf attribute combinations; $\boldsymbol{v}, \boldsymbol{f}, \boldsymbol{a}$ are respectively vectors of real, forecast and expected values of all affected **leaf** attribute combinations, *i.e.*, $\boldsymbol{v}(S){=}(v(e_1), v(e_2), ..., v(e_n))$; and $\text{avg}(\cdot)$ means average over the vector.

The most important difference between GPS (3) and *potential score* ($ps{=}\max(0, 1{-}\frac{||(v(S_1),v(S_2)){-}(a(S_1),f(S_2))||_2}{||(v(S_1),v(S_2)){-}(f(S_1),f(S_2))||_2})$) in [7] is that we use the sum of **normalized** L1-norm of the positive and negative parts, while *potential score* uses L2-norm of all leaf attribute combinations. We make this modification because the forecast residuals of normal part would cumulate as the number of normal leaf attribute combinations increases.

An example is given in Table I. The two bold lines are abnormal. The other normal lines may have small forecast errors compared to the normal values. Thus, *potential score* gives a very low score for the root cause (Province=Beijing), while GPS gives a high score. This demonstrate GPS' advantage over potential score.

$$ps = 1 - \frac{||(0, 0, 1, 0.2, 0, 10, 2, 3, 2)||_2}{||(5, 10, 1, 0.2, 0, 10, 2, 3, 2)||_2} = 0.30$$

$$GPS = 1 - \frac{\text{avg}(0, 0) + \text{avg}(1, 0.2, 0, 10, 2, 3, 2)}{\text{avg}(5, 10) + \text{avg}(1, 0.2, 0, 10, 2, 3, 2)} = 0.74$$

Therefore, when the magnitude of anomaly is not significant, potential score will fail to indicate the root cause because it cumulates forecast errors of all leaf attribute combinations.

In each cuboid, we would get a root cause candidate. If its corresponding GPS is larger than a threshold δ, *Squeeze* would not search deeper layer. Note that δ is the only parameter in *Squeeze* that has to manually configured, and in Section VI-E we show *Squeeze* is very robust against different δ values.

Finally all the root cause candidates will be sorted by their GPS and succinctness. Succinctness is defined as the number of attributes of all root cause attribute combinations. We use a constant C to trade off between GPS and succinctness. We use an empirical formula for C, which avoids manual tuning:

$$-\frac{\log(\#clusters \cdot \frac{\#abnormal\ leaves}{\#leaves})}{\log(\#attribute\ values)} \cdot (\#attribute\ values)$$

V. Success Stories in Industrial Practice

We have successfully applied *Squeeze* to several large commercial banks and a top Internet company. The results show that *Squeeze* can localize root causes much more rapidly and accurately than traditional manual analysis. In this section, we present three success stories. We omit and anonymize some confidential details.

A. Case I: Intra-System localization

In 26 Nov 2018 9:00 to 11:00, the operators of a bank received many tickets and alerts and noticed that the API call success rate of a system suffered a severe drop. The search space is large, and the attributes and the number of distinct values are listed as follows: province (38), agency (815), server group (16), channel (4), server (339), code (4), status (2), service type (3). After two hours of fruitless manual root cause localization, the operators decided to just roll back the entire system to the last version, which happened to actually fix the issue. After the roll-back, it took another 2 hours for an inexperienced operator on duty to eventually find the root cause (there was a bug in the newly deployed version of the software for ServiceType 0200020) based on the 2-hour logs during the service anomaly.

Upon the request of the operators, we ran *Squeeze* over this system's logs at the beginning of the service anomaly. *Squeeze* took a few seconds to report the root cause (ServiceType=020020), which indicates exactly the software that has a buggy version update. Had *Squeeze* be actually used immediately after the anomaly happened, operators could have localized the root cause much faster.

Fig. 5 shows this case's results of deviation-based clustering. We can see that the deviation scores of all leaf attribute combinations of the root cause (ServiceType=020020) are very close to each other. This to some extent confirms the generalized ripple effect.

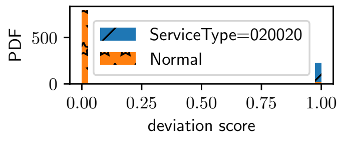

Fig. 5: The histogram of deviation scores of leaf attribute combinations in *Case I*.

B. Case II: Inter-System localization

In 23 Mar 2019, there was a burst of failures in another bank's transaction system. There are many subsystems which communicate with each other by API calls. The search space is also large, and the attributes and the number of distinct values are listed as follows: source (13), source ip (66), destination (7), destination ip (10), interface (135). The operators manually located *one* root cause (destination=ic in Fig. 6) in ten minutes by tracing all the abnormal transactions' API calls.

Again, upon the request of the operators, we ran *Squeeze* over this system's API call logs at the beginning of the service anomaly. *Squeeze* localized the root causes in Fig. 6 in just several seconds, which also confirms the generalized ripple effect because deviation scores of leaf attribute combinations that are descended from the same root cause attribute combination are close to each other. Note that *Squeeze* reported more root cause attribute combinations than what the operators found. The operators confirmed that these additional root causes are indeed valid: they were abnormal and were actually affected by ic system. Had *Squeeze* be actually used immediately after the issue, operators could have localized the root cause much faster (seconds vs minutes) and more accurately. We also run some other algorithms on Case II (see Table VI).

TABLE VI: Qualitative comparison on industrial cases: whether the algorithm can find the true root cause. Some cases and some baseline algorithms are missing due to deployment issues. Details about the algorithms are in Section VII.

RC?	*Squeeze*	HotSpot	Apriori	Adtributor	R-Adtributor
Case II	Yes	Yes	Yes	No	No
Case III	Yes	No	No	No	No

C. Case III: Anomaly with Insignificant Magnitude

One day night, a top Internet company suffered an anomaly. The HTTP error counts bursted suddenly as shown in Fig. 7. The attributes and the number of distinct values are listed as follows: datacenter (11), province (7), ISP (6), useragent(22). The operators manually found a potential root cause which consists of only one attribute combination ($AC1$ in Fig. 7), which took them one hour.

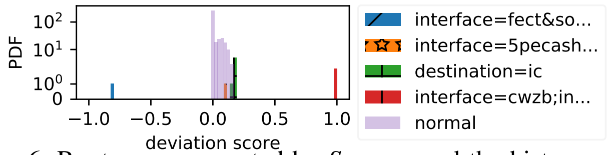

Fig. 6: Root causes reported by *Squeeze* and the histogram of deviation scores of their descent leaves in *Case II*.

We ran *Squeeze* over this system's logs at the beginning of the service anomaly and in several seconds found more root causes: $AC1$ and $AC2$ as shown in Fig. 7. It is obvious that $AC2$ also has severe error bursts. The operators mistakenly ignored it because the error count of $AC2$ only represented a very small fraction of the total error counts. Manual analysis apparently has difficulties in localizing root causes of anomalies with insignificant magnitudes. *Squeeze* would help the operators to notice such root causes efficiently.

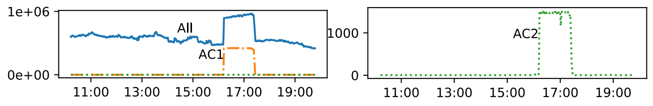

Fig. 7: Measure values along time of *Case III*. $AC1$ is (useragent=uc$\wedge$idc=ih$\wedge$province=other), and $AC2$ is (useragent=uc$\wedge$idc=is$\wedge$province=other). Note that the Y-axis scales are different.

VI. Experiments

In this section, we use the results on the semi-synthetic datasets to answer these three questions:

- RQ1: How effective is *Squeeze* in localizing root causes? Is it effective for both fundamental and derived measures? Is it effective for anomalies with any magnitudes?
- RQ2: How efficient is *Squeeze* in localizing root causes? It is efficient consistently for all cases?
- RQ3: How well does *Squeeze* perform in different configurations? Is *Squeeze* sensitive to parameters?

We compare *Squeeze* with the following previous works, which will be introduced with more details in Section VII: HotSpot [7] (we revise it to handle derived measures according to GRE in Section III), iDice [5], Adtributor [3], R-Adtributor [4], Apriori [6].

A. Evaluation Metrics

F1-score is used in this paper to evaluate multi-dimensional root cause localization tasks. It is calculated based on attribute combinations, *i.e.*, if an attribution combination is reported by the algorithm and it is in the root cause, it is a true positive (TP); if a reported attribute combination is not in the root cause, it is a false positive (FP); if an attribute combination in root cause is not reported, it is a false negative (FN). Then F1-score is calculated as follows: $F1\text{-}Score = \frac{2*\#TP}{2*\#TP+\#FP+\#FN}$.

Another evaluation metric is time cost. In the following experiments, we present the average running time of all cases in the corresponding setting.

B. Datasets

We have two real-world datasets from two companies, which are collected from real systems. One of them is an online shopping platform, and the other one is an Internet company. They are denoted by $\mathcal{I}_1, \mathcal{I}_2$ respectively. The measure of $\mathcal{I}_1$ is the number of transactions every five minutes, and the measure of $\mathcal{I}_2$ is the number of page views per minute. There are 5 attributes in $\mathcal{I}_1$ and 4 in $\mathcal{I}_2$. Although these are real data, it is very hard to get lots of anomalies and corresponding root causes verified by operators manually as the ground truth. As a result, we inject synthetic anomalies onto these real data to evaluate the algorithm. We get 7 semi-synthetic datasets with different anomaly injection methods based on $\mathcal{I}_1$ and $\mathcal{I}_2$[1]. The basic statistics of them are shown in Table VII, where d means the number of attributes, #AC means the number of leaf attribute combinations, and n means the number of injected anomalies for **each setting**. In each setting, a root cause contains the same number of root cause attribute combinations in cuboids of the same layer.

TABLE VII: Summary of Datasets

Name	n	d	#AC	Source	Measure	Residual
$\mathcal{A}$	400	5	15324	$\mathcal{I}_1$	Fundamental	13.0%
$\mathcal{B}_0$	100	4	21600	$\mathcal{I}_2$	Fundamental	0.80%
$\mathcal{B}_1$	100	4	21600	$\mathcal{I}_2$	Fundamental	3.19%
$\mathcal{B}_2$	100	4	21600	$\mathcal{I}_2$	Fundamental	6.37%
$\mathcal{B}_3$	100	4	21600	$\mathcal{I}_2$	Fundamental	9.54%
$\mathcal{B}_4$	100	4	21600	$\mathcal{I}_2$	Fundamental	13.0%
$\mathcal{D}$	100	4	21600	$\mathcal{I}_2$	Derived	3.99%

For fundamental measures, we randomly choose some cuboids and root causes in them, change the values of the abnormal leaf attribute combinations by GRE with random amounts, finally add Gaussian noises to all leaf attribute combinations. We use the Gaussian noises with different standard deviations to emulate different forecast residuals.

For derived measures ($\mathcal{D}$), we assume normal success rates follow $unif(0.9, 1.0)$, pick some attribute combinations, then make their success rate decrease according to GRE, then randomly generate the total transaction number and successful transaction number, finally add Gaussian noises with standard deviation 5% to all attribute combinations.

In all datasets, we inject anomalies with random magnitudes, *i.e.*, we do not guarantee the anomalies are significant.

We apply MA (moving average) for all scenarios. We present the time usage for single leaf attribute combination of several algorithms used by previous works (Period is used by [7], ARIMA is used by [3]) in Table VIII. MA is one of the simplest forecast algorithms and it costs little time.

TABLE VIII: Time Usage Comparison of Forecast Methods

Algorithm	MA	Period [19]	ARIMA
Time Usage (μs)	6.11($\pm$1.91)	29.8($\pm$3.94)	38575($\pm$32447)

C. RQ1: Effectiveness

We conduct experiments to evaluate *Squeeze* with datasets described in Section VI-B. We present the F1-score comparison of fundamental measures in Table IX, and present that of

[1]Datasets and code are available at https://github.com/lizeyan/Squeeze

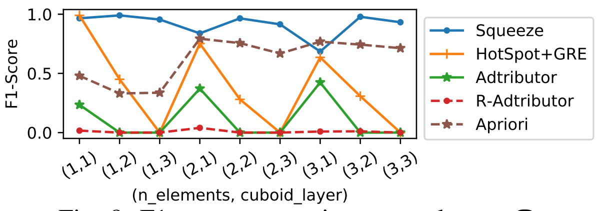

Fig. 8: F1-score comparison over dataset $\mathcal{D}$.

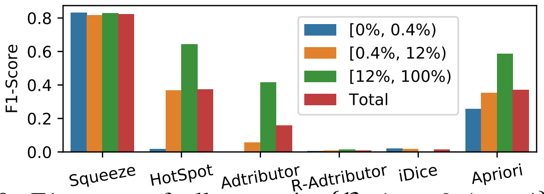

Fig. 9: F1-scores of all cases in $\{\mathcal{B}_i, i = 0, 1, ..., 4\}$ with different abnormal magnitudes.

$\mathcal{D}$ (derived measure) in Fig. 8, where *cuboid_layer* means the layer of the cuboid contains the root causes and *n_elements* means the number of attribute combinations in the root causes. We present the total F1-score of each setting in Table IX and Fig. 8. We set $\delta = 0.9$ for all cases. Other algorithms' parameters are set following the original papers' suggestions.

For all cases, *Squeeze* is able to achieve relatively good results. In most cases, *Squeeze* achieves the best F1-scores. iDice does not work on our data mainly because it is designed for detecting anomalies and localizing root causes for a duration, rather than a single time point. Adtributor only localizes root causes of the first-layer cuboids, and the *explanation power* makes it only able to handle anomalies with significant magnitudes. Although R-Adtributor is able to localize root causes in other cuboids, it is hard for R-Adtributor to decide when the recursion should terminate. HotSpot has the problem in localizing root causes which are in deep cuboids or containing many attribute combinations. HotSpot relies on *potential scores* along the whole search path to lead it to the true root cause, but for root causes which are in deep cuboids or containing many attribute combinations, the *potential scores* can be quite small at the beginning of the search path. As a result, on the one hand, the *hierarchical pruning strategy* may wrongly prune the right search path. On the other hand, for the sake of efficiency, HotSpot cannot search too many steps and therefore the mistakes at the beginning of search can be fatal for HotSpot. Furthermore, we revise HotSpot with GRE to obtain HotSpot+GRE so that it can handle derived measures.

Squeeze performs well for anomalies with any magnitudes. We present the total F1-score of $\mathcal{B}_0, \mathcal{B}_1, ..., \mathcal{B}_4$ of different levels of magnitudes in Fig. 9. *Squeeze*'s F1-score outperforms others by about **0.4** in $\{\mathcal{B}_i, i = 0, 1, ..., 4\}$ in total, as shown in Fig. 9. $[0\%, 0.4\%)$ means the magnitudes of anomalies are less than 0.4% of the sum of all leaf attribute combinations, and so do the other two. We choose 0.4% and 12% because they are 25-percentile and 75-percentile respectively. There is almost nothing different for *Squeeze* whether the anomaly magnitude is significant or not. Only *Squeeze* achieves the same good performance in all cases. Hotspot and Adtributor are almost not able to localize root causes for insignificant

TABLE IX: F1-score comparison of fundamental measures

F1-score		(n_elements, cuboid_ayer)								
Dataset	Algorithm	(1, 1)	(1, 2)	(1, 3)	(2, 1)	(2, 2)	(2, 3)	(3,1)	(3,2)	(3, 3)
$\mathcal{A}$	*Squeeze*	**0.8632**	**0.7827**	**0.4932**	**0.7584**	**0.6361**	**0.4097**	**0.6441**	**0.5145**	**0.3618**
	HotSpot	0.6856	0.4389	0.2158	0.5085	0.3433	0.2043	0.3988	0.2916	0.1768
	Adtributor	0.3892	0.0000	0.0000	0.4010	0.0000	0.0000	0.3857	0.0000	0.0000
	R-Adtributor	0.0180	0.0020	0.0016	0.0075	0.0049	0.0294	0.0081	0.0067	0.0410
	iDice	0.0000	0.0036	0.0425	0.0000	0.0065	0.0437	0.0000	0.0007	0.0172
	Apriori	0.1036	0.0580	0.0001	0.1427	0.0926	0.0019	0.1537	0.0882	0.0062
$\mathcal{B}_0$	*Squeeze*	0.9041	**0.9327**	**0.9231**	**0.9604**	**0.9799**	**0.9333**	**0.9631**	**0.9371**	**0.9228**
	HotSpot	**0.9950**	0.4928	0.1215	0.7588	0.3934	0.0577	0.5961	0.3043	0.0775
	Adtributor	0.3044	0.0000	0.0000	0.4226	0.0000	0.0000	0.4654	0.0000	0.0000
	R-Adtributor	0.0639	0.0000	0.0000	0.0114	0.0000	0.0000	0.0177	0.0000	0.0000
	iDice	0.0000	0.0517	0.0488	0.0000	0.0409	0.0618	0.0000	0.0228	0.0959
	Apriori	0.4430	0.5116	0.7523	0.8490	0.6853	0.7351	0.8743	0.8087	0.7368
$\mathcal{B}_1$	*Squeeze*	0.8900	**0.8889**	**0.8350**	**0.9624**	**0.9479**	**0.9192**	**0.8912**	**0.9019**	**0.9060**
	HotSpot	**0.9804**	0.3767	0.0175	0.6882	0.4207	0.0549	0.6030	0.3261	0.0471
	Adtributor	0.2874	0.0000	0.0000	0.4286	0.0000	0.0000	0.5000	0.0000	0.0000
	R-Adtributor	0.0538	0.0000	0.0081	0.0114	0.0000	0.0000	0.0100	0.0000	0.0000
	iDice	0.0000	0.0040	0.0065	0.0000	0.0125	0.0099	0.0000	0.0245	0.0092
	Apriori	0.6965	0.6582	0.6316	0.8492	0.8351	0.8161	0.8885	0.7986	0.0000
$\mathcal{B}_2$	*Squeeze*	0.8155	**0.8208**	**0.9171**	**0.9641**	**0.8021**	**0.8615**	0.7081	0.7731	**0.8396**
	HotSpot	**0.9167**	0.3930	0.0254	0.6648	0.3486	0.0601	0.5839	0.3349	0.0563
	Adtributor	0.2915	0.0000	0.0000	0.4590	0.0000	0.0000	0.4848	0.0000	0.0000
	R-Adtributor	0.0100	0.0000	0.0000	0.0201	0.0000	0.0041	0.0177	0.0000	0.0034
	iDice	0.0000	0.0063	0.0030	0.0000	0.0119	0.0123	0.0000	0.0125	0.0108
	Apriori	0.3844	0.3208	0.1537	0.8184	0.7556	0.2636	**0.8885**	**0.8345**	0.3339
$\mathcal{B}_3$	*Squeeze*	0.8835	**0.7611**	**0.6920**	**0.8667**	**0.7809**	**0.7494**	0.6753	0.7068	**0.7586**
	HotSpot	**0.9167**	0.3810	0.0339	0.6744	0.3988	0.0299	0.5369	0.4072	0.0625
	Adtributor	0.2770	0.0000	0.0000	0.4182	0.0000	0.0000	0.4610	0.0000	0.0000
	R-Adtributor	0.0000	0.0000	0.0052	0.0086	0.0000	0.0188	0.0177	0.0000	0.0226
	iDice	0.0000	0.0061	0.0025	0.0000	0.0180	0.0129	0.0000	0.0189	0.0175
	Apriori	0.0000	0.5000	0.2792	0.0000	0.7324	0.4605	**0.8825**	**0.8330**	0.5314
$\mathcal{B}_4$	*Squeeze*	0.8173	**0.7207**	**0.6167**	**0.8528**	**0.6863**	**0.7015**	**0.5470**	**0.6489**	**0.6543**
	HotSpot	**0.9346**	0.3362	0.0084	0.6535	0.3642	0.0301	0.5282	0.3431	0.0321
	Adtributor	0.2740	0.0000	0.0000	0.3853	0.0000	0.0000	0.4517	0.0000	0.0000
	R-Adtributor	0.0000	0.0000	0.0037	0.0143	0.0000	0.0092	0.0301	0.0000	0.0233
	iDice	0.0000	0.0092	0.0049	0.0000	0.0069	0.0164	0.0000	0.0229	0.0108
	Apriori	0.0467	0.0064	0.0000	0.1290	0.0042	0.0016	0.2023	0.0093	0.0009

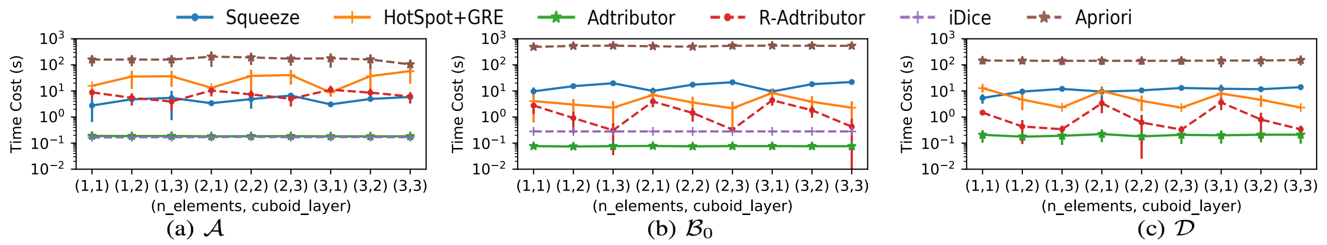

Fig. 10: Running time comparison of $\mathcal{A}, \mathcal{B}_0, \mathcal{D}$. Mean and standard deviation of each setting are presented.

cases, mainly because of their *potential score* and *explanation power* mechanisms. Apriori is able to localize root causes for anomalies with insignificant magnitudes, but it may prune such anomalies because of its sensitive *support* threshold. Therefore its performance on such cases is not so good.

D. RQ2: Efficiency

Squeeze is efficient enough in all cases. In Fig. 10 we present the running times of $\mathcal{A}, \mathcal{B}_0, \mathcal{D}$. We do not present results of all datasets because all of $\{\mathcal{B}_i, i = 0, 1, 2, 3, 4\}$ have similar results. *Squeeze* costs only about ten seconds even in the worst cases. It is efficient enough since measures are usually collected every one minute or every five minutes. HotSpot sometimes is as efficient as *Squeeze*, but sometimes it would cost more time. Apriori costs hundreds of seconds, which is too slow. Others can be fast, but they do not effectively localize root causes. We run every experiment on a server with 24 × Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz (2 sockets) and 64G RAM. All algorithms are implemented with Python but iDice is with C++. Experiments of all algorithms are conducted under the same condition.

E. RQ3: Performance under Different Configurations

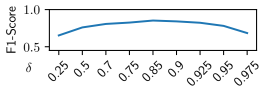

Fig. 11: F1-Scores over different δ of $\mathcal{B}_1$ with $cuboid\_layer = 3, n\_elements = 3$.

All the parameters in Section IV are automatically configured except δ, the GPS threshold. We present *Squeeze*'s effectiveness under different δ in Fig. 11 with $cuboid\_layer = 3, n\_elements = 3$ in $\mathcal{B}_1$. *Squeeze*'s performance does not change a lot as δ changes. We only choose this setting because it is the hardest setting, and actually results in Table IX and Fig. 8 also show that $\delta = 0.9$ leads to a good enough results. Since δ is the threshold of GPS for early stopping, it is reasonable to set δ near 0.9 regardless of the specific dataset.

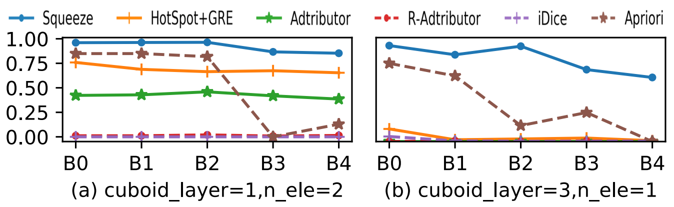

Fig. 12: F1-score comparison of different residuals.

Squeeze, as well as many other algorithms, relies on forecast values. In Fig. 12, we plot the F1-scores under different forecast residuals. We select these two representative settings because *Squeeze*'s performance changes relatively small in (a) and large in (b) along different datasets. Residuals do affect the performance of algorithms that rely on forecasting including *Squeeze*. But *Squeeze* still outperforms other algorithms when the residuals are relatively large.

F. Thread to Validity

The experiments in this section are based on semi-synthetic datasets, which might not perfectly represent real world ones.

VII. Related Works

There are many previous works in root cause localization in various contexts. Most works are different from ours [20]–[31]. On the one hand, we focus on root cause localization on general multi-dimensional systems. On the other hand, most of these works use intuitive domain-knowledge based empirical methods, while we propose a generic algorithm. Some works are closely related to ours. We compared them in Table III.

Adtributor [3] assumes that the root cause can be localized by only one attribute. It uses different approaches for fundamental and derived measures. Adtributor relies on forecasting for attribute combinations in all one-attribute cuboids to calculate *explanation power* and *surprise*. R-Adtributor [4] recursively calls Adtributor to provide multi-dimensional root causes. But it has a hard-to-tune termination condition.

iDice [5] only works for fundamental measures and anomalies with significant magnitudes. Isolation power measures how well the attribute combination isolates abnormal and normal values, but if there are more than one attribute combination related to anomalies, a single attribute combination actually cannot isolate abnormal and normal values.

HotSpot [7] uses MCTS (Metro Carlo tree search) to solve the problem of huge search space. It focuses on fundamental measures since they consider ripple effect to hold only for fundamental measures. HotSpot relies on forecasting for all leaf attribute combinations to calculate *potential score*. As shown in Section IV-D, *potential score* does not work well when anomaly magnitudes are not very significant.

End2End [6] uses forecast value and real value to distinguish all abnormal leaf attribute combinations. Then the Apriori algorithm is applied to find the max frequent patterns related to anomalies with high confidence. [6] mainly focuses on accurately and efficiently forecasting for all leaf attribute combinations. The appropriate thresholds of support, confidence and distinguishing normal and abnormal leaf attribute combinations are different for different services or the service over time. Apriori's performance is sensitive to these thresholds. So its performance varies in different settings, as Section VI shows.

VIII. Limitations and Future Works

Squeeze assumes that attribute combinations of the root cause which cause the same abnormal magnitudes (in the same cluster in Section IV-C2) are in one cuboid. This assumption is very weak since in practice it is very rare that root cause attribute combinations which cause the same abnormal magnitudes are in more than one cuboid.

Squeeze focuses on categorical attributes, and *Squeeze* cannot leverage numerical attributes. We observe that numerical attributes are much less prevalent in practice (e.g. in the four companies that we studied in this paper). According to our interviews with some engineers, they choose not to record them because they weren't sure how to use them for diagnosis.

In Section III, we only give proof of quotients. We have also proven GRE for products. It is omitted due to space limitations, and it is similar to that of quotients. They are the most common two cases. More types of derived measures' GRE can be proved when we encounter them. In most cases, the idea of *finite difference* would help. But for those which are not continuous (*e.g.*, 95-percentile of search response time), it is more challenging. We will work on it in the future.

We apply a relatively simple clustering algorithm in Section IV-C2, and we use the *deviation based filtering* to make the clustering perform better. Since it is simple, an advanced clustering algorithm may make *Squeeze* work better.

IX. Conclusion

Given the importance of multi-dimensional root cause localization, many approaches are proposed, but they are not generic or robust enough and suffer from some limitations. Through a novel "bottom-up then top-down" searching strategy and the techniques based on our proposed generalized ripple effect and generalized potential score, our proposed algorithm, *Squeeze*, overcomes these limitations. *Squeeze* achieves a good trade-off between speed and accuracy in a generic and robust manner. Case studies in several large commercial banks and an Internet company show that *Squeeze* can localize root causes much more rapidly and accurately than traditional manual analysis. Furthermore, we conduct extensive experiments on several semi-synthetic datasets. The results show that the F1-score of *Squeeze* outperforms previous approaches by 0.4 on average, and consistently costs only about 10s.

X. Acknowledgment

The authors gratefully thank Suning.com Co., Ltd. for providing parts of the data used in this paper's evaluation. We thank Juexing Liao and Chuanxi Zheng for proofreading this paper. This work has been supported by the Beijing National Research Center for Information Science and Technology (BNRist) key projects.

References

[1] 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 *International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2015, pp. 1–13.

[2] M. Ma, S. Zhang, D. Pei, X. Huang, and H. Dai, "Robust and rapid adaption for concept drift in software system anomaly detection," in *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2018, pp. 13–24.

[3] R. Bhagwan, R. Kumar, R. Ramjee, G. Varghese, S. Mohapatra, H. Manoharan, and P. Shah, "Adtributor: Revenue debugging in advertising systems," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, 2014, pp. 43–55.

[4] M. Persson and L. Rudenius, "Anomaly detection and fault localization an automated process for advertising systems," Master's thesis, 2018, göteborg : Chalmers University of Technology.

[5] Q. Lin, J.-G. Lou, H. Zhang, and D. Zhang, "idice: problem identification for emerging issues," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 214–224.

[6] 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 Transactions on Networking (TON)*, vol. 25, no. 6, pp. 3709–3722, 2017.

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

[8] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[9] F. Knorn and D. J. Leith, "Adaptive kalman filtering for anomaly detection in software appliances," in *INFOCOM Workshops 2008, IEEE*. IEEE, 2008, pp. 1–6.

[10] B. Pincombe, "Anomaly detection in time series of graphs using arma processes," *Asor Bulletin*, vol. 24, no. 4, p. 2, 2005.

[11] Y. Chen, R. Mahajan, B. Sridharan, and Z.-L. Zhang, "A provider-side view of web search response time," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 243–254.

[12] S. Zhang, Y. Liu, D. Pei, Y. Chen, X. Qu, S. Tao, Z. Zang, X. Jing, and M. Feng, "Funnel: Assessing software changes in web-based services," *IEEE Transactions on Service Computing*, 2016.

[13] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2018, pp. 187–196.

[14] Z. Li, W. Chen, and D. Pei, "Robust and unsupervised kpi anomaly detection based on conditional variational autoencoder," in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2018, pp. 1–9.

[15] J. Bu, Y. Liu, S. Zhang, W. Meng, Q. Liu, X. Zhu, and D. Pei, "Rapid deployment of anomaly detection models for large number of emerging kpi streams," in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2018, pp. 1–8.

[16] 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 *Proceedings of the 2015 Internet Measurement Conference*. ACM, 2015, pp. 211–224.

[20] M. Y. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer, "Pinpoint: Problem determination in large, dynamic internet services," in *Proceedings International Conference on Dependable Systems and Networks*. IEEE, 2002, pp. 595–604.

[17] L. M. Milne-Thomson, *The calculus of finite differences*. American Mathematical Soc., 2000.

[18] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a" kneedle" in a haystack: Detecting knee points in system behavior," in *2011 31st International Conference on Distributed Computing Systems Workshops*. IEEE, 2011, pp. 166–171.

[19] S.-B. Lee, D. Pei, M. Hajiaghayi, I. Pefkianakis, S. Lu, H. Yan, Z. Ge, J. Yates, and M. Kosseifi, "Threshold compression for 3g scalable monitoring," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 1350–1358.

[21] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, "Detection and localization of network black holes," in *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE, 2007, pp. 2180–2188.

[22] S. Kandula, D. Katabi, and J.-P. Vasseur, "Shrink: A tool for failure diagnosis in ip networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*. ACM, 2005, pp. 173–178.

[23] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, "Fault localization via risk modeling," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 396–409, 2009.

[24] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang, "Towards highly reliable enterprise network services via inference of multi-level dependencies," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 13–24, 2007.

[25] 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 *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 2756–2760.

[26] B. Nguyen, Z. Ge, J. Van der Merwe, H. Yan, and J. Yates, "Absence: Usage-based failure detection in mobile networks," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 2015, pp. 464–476.

[27] D. Liu, Y. Zhao, K. Sui, L. Zou, D. Pei, Q. Tao, X. Chen, and D. Tan, "Focus: Shedding light on the high search response time in the wild," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[28] S. Zhang, W. Meng, J. Bu, S. Yang, Y. Liu, D. Pei, J. Xu, Y. Chen, H. Dong, X. Qu *et al.*, "Syslog processing for switch failure diagnosis and prediction in datacenter networks," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*. IEEE, 2017, pp. 1–10.

[29] S. Zhang, Y. Liu, W. Meng, Z. Luo, J. Bu, S. Yang, P. Liang, D. Pei, J. Xu, Y. Zhang *et al.*, "Prefix: Switch failure prediction in datacenter networks," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 1, p. 2, 2018.

[30] W. Meng, Y. Liu, S. Zhang, D. Pei, H. Dong, L. Song, and X. Luo, "Device-agnostic log anomaly classification with partial labels," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 2018, pp. 1–6.

[31] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, and R. Zhou, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 4739–4745.