HotSpot: Anomaly Localization for Additive KPIs with Multi-Dimensional Attributes

Yongqian Sun[†], Youjian Zhao[†], Ya Su[†], Dapeng Liu[§], Xiaohui Nie[†], Yuan Meng[†],
Shiwen Cheng[†], Dan Pei[†*], Shenglin Zhang[‡], Xianping Qu[§], Xuanyou Guo[§]
[†]Tsinghua National Laboratory for Information Science and Technology (TNList) & Tsinghua University
[‡]Nankai University, [§]Baidu
{sunyq12, sy-16, nxh15, meng-y16, csw14}@mails.tsinghua.edu.cn,
{zhaoyoujian, peidan}@tsinghua.edu.cn, zhangsl@nankai.edu.cn,
{liudapeng01, quxianping, guoxuanyou01}@baidu.com

Abstract—Additive key performance indicators (KPIs, such as page view, revenue, error count) with multi-dimensional attributes (such as ISP, Province, DataCenter) are common and important monitoring metrics in Internet companies. When an anomaly happens to an overall KPI, it is critical but challenging to localize the root cause, which is one (or more) combination of attribute values in multiple dimensions. For example, is the total page view (PV) decrease caused by the PV decrease from "Beijing" or "China Mobile in Beijing", or "Beijing and Shanghai"? However, this task is very challenging for two major reasons. First, the PVs of different combinations are interdependent, thus the PV anomalies at the root cause can cause the changes of many other PVs at different aggregation levels. Second, there could be tens of thousands of combinations to investigate in multi-dimensional attribute space. It is a difficulty to find the root cause from a huge search space.

To address the first challenge, our approach HotSpot uses a novel potential score based on the ripple effect for anomaly propagation that we reveal. To address the second challenge, HotSpot adopts the Monte Carlo Tree Search (MCTS) algorithm and a hierarchical pruning strategy. Using the real-world data from a top global search engine, we show that HotSpot achieves a great improvement on effectiveness and robustness, i.e., 95% of all types of root cause cases using HotSpot (compared to only 15% using existing approaches) achieves a F-score over 90%. Operational experiences show that HotSpot can reduce the localization time from more than 1 hour in manual efforts to less than 20 seconds.

Index Terms—Anomaly localization, multi-dimensional attributes, huge search space, potential score, Monte Carlo Tree Search (MTCS), hierarchical pruning.

I. Introduction

To provide good quality of service, Internet companies all monitor a collection of key performance indicators (KPIs), among which additive KPIs (such as page view, revenue, traffic volume) with multi-dimensional attributes are common and important ones. For example, page view (PV) of a website (the number of user accesses per time interval) is closely related to the website's revenue, thus should be closely monitored

The KPI records can have several attributes, such as Province (the geo-region mapped from the user's IP), ISP (user's access ISP), DC (the data center where the request is served). Each attribute has a range of distinct values. Generally, we record the ***KPI values*** in every time interval (e.g., every minute) for each distinct combination of the attribute values, e.g., $(Beijing, ChinaTelecom, DC1)$. These most fine-grained KPI records, thanks to the KPI's additive nature, can be naturally summed up into more coarse-grained KPIs. For example, all the KPI records with $Province = Beijing$ and $ISP = ChinaTelecom$ regardless of $DataCenter$ can be summed up into $(Beijing, ChinaTelecom, *)$, where $*$ is a wildcard.

When an anomaly, e.g., a sudden increase or decrease, happens to a total KPI (i.e., for (*,*,*)), it is critical but challenging to quickly localize the root cause, i.e., the elements which have the most potential to have caused the total KPI anomaly, so that operators can take actions to mitigate the problem. Note that in this paper we only deal with the case where the total KPI value is anomalous.

The root cause can be one (or more) combination of attribute values in multiple dimensions. Thus, the major challenge for root cause localization is the huge search space for potential root causes. First, the changes in the root cause, e.g., $(Beijing, ChinaTelecom, *)$, can propagate to more coarse-grained combinations, e.g., $(Beijing, *, *)$, $(*, ChinaTelecom, *)$, more fine-grained combinations, e.g., $(Beijing, ChinaTelecom, DC1)$, through which other related combinations, e.g., $(*, *, DC1)$ are also impacted. Second, the root cause can be a set of multiple values of the same attribute e.g., $(Beijing\ and\ Tianjin\ and\ Hebei, *, *)$ (where these three geographically adjacent provinces are impacted by a same failure). The number of such combinations is huge, e.g., the number of combinations of various province values in China is $2^{36} - 1$.

While the attribute combinations of real-world root cause cases can be very complex, existing works such as Adtributor [1] and iDice [2] can only deal with simple cases with much smaller search space (see §V-F for more details). This paper proposes an approach, called HotSpot, to automatically localize the root cause, one (or more) combination of attribute values, that has made the total value anomalous for an additive KPI with multi-dimensional attributes. The main contributions of this

Dan Pei[†*] is the corresponding author.

TABLE I: Terms

Term	Definition	Notation	Example
Attributes	The categories of the information of each PV record	—	Province(P), ISP(I), DC(D), Channel(C)
Attribute values	The candidate values of each attribute	—	{Beijing, Shanghai, Guangdong} for Province(P)
Element	A combination vector of distinct values of each attribute	$e=(p,i,d,c)$	(Beijing,*,*,*), (*,Mobile,*,*), (Beijing, Mobile,*,*)
PV value	The number of access logs according to an element	$v(e_i)$	v(Beijing,*,*,*)
Forecast value	Forecast PV value of an element using the historical values	$f(e_i)$	f(Beijing,*,*,*)
Data cube	A data structure of multi-dimensional data	*n-d cube*	A 4-d data cube with the dimensions {P,I,D,C}
Cuboids	A cuboid is a data cube whose dimensions are in a subset of all given dimensions	B_i	$\{B_P$, $B_{P,I}$, $B_{P,I,D}$,...} for the 4-d data cube with the dimensions {P,I,D,C}
Potential Score	A concept of measuring the potential of a set of elements to be the root cause	ps	ps(S), S={(Beijing,*,*,*), (*,Mobile,*,*)}

paper are summarized as bellow:

- To deal with the huge search space of root causes, HotSpot adopts the MCTS approach (the first time in anomaly localization literature).
- The action value in adopting MCTS is our novel potential score based on the "ripple effect", which captures how the change of the KPI value for one attribute combination (as a cause) can cause other attribute combinations' KPI values change (as effects) for multi-dimensional additive KPIs.
- We propose a hierarchical pruning approach (similar to the Apriori Principle in spirit) to further reduce the search space.
- Using the real-world data from a top global search engine, we show that HotSpot achieves a great improvement when compared with two existing approaches both on effectiveness and robustness, i.e., HotSpot achieves F-score over 90% for 95% of all types of cases, while for existing approaches only less than 15% of all types of cases have a F-score over 90%.
- Our operational experiences show that HotSpot can reduce the localization time from more than 1 hour in manual efforts to less than 20 seconds.

The rest of this paper is organized as follows. In Section 2, we present the problem statement of anomaly localization. We show the core idea and the overview of HotSpot in Section 3, and then present the design of HotSpot in Section 4. In Section 5, we evaluate the performance of HotSpot using experiments driven by real-world data. In Section 6, we present the operational experience of HotSpot. Discussion, related work and conclusions are presented in Sections 7, 8 and 9, respectively.

II. Problem definition

We first introduce some terminologies (summarized in Table I) in our paper, and then present the problem statement of anomaly localization and its challenges. Without loss of generality, throughout the paper, we will use PV as our primary example of additive KPI, and Province, ISP, Data Center, Ad Channel (An ad attribute that reflecting the positions), including cardinalities, as example attributes. Note that these are examples for better presentation clarity.

A. Important terms

A PV record at the website can have several attributes. For example, "10:00:01 (*Timestamp*); *Beijing*, *Mobile*, DC_1, $Channel_1$" is a record, and Beijing, Mobile, DC_1 and $Channel_1$ are the candidate values according to four attributes respectively, i.e., Province (P), ISP (I), Data Center (D) and Channel (C), where $\mathrm{P}=\{p\}$, $\mathrm{I}=\{i\}$, $\mathrm{D}=\{d\}$, $\mathrm{C}=\{c\}$ are the set of 36, 10, 6, 10 distinct values of province, ISP, data center, and ads channel, respectively. The values of P and I are based on the client IP and resolved by using a IP-to-geolocation database and BGP table, respectively. Each ISP at each province is a standalone company, thus the same ISP names at different provinces often behave differently. Channels are the labels for different ad markets, e.g., medical or education. Table II shows some examples of PV records.

TABLE II: PV records

Timestamp; P,I,D,C
10:00:01; Beijing, Mobile, DC_1, $Channel_1$
10:00:01; Beijing, Mobile, DC_1, $Channel_2$
10:00:12; Beijing, Unicom, DC_1, $Channel_2$
10:00:30; Beijing, Unicom, DC_1, $Channel_2$
10:00:45; Beijing, Mobile, DC_1, $Channel_1$
10:00:59; Beijing, Unicom, DC_1, $Channel_2$
10:01:03; Beijing, Mobile, DC_1, $Channel_1$
...

TABLE III: Elements and PV values

Time	element (p,i,d,c)	PV Value
10:00	(Beijing, Mobile, DC_1, $Channel_1$)	2
10:00	(Beijing, Mobile, DC_1, $Channel_2$)	1
10:00	(Beijing, Unicom, DC_1, $Channel_2$)	3
10:01	(Beijing, Mobile, DC_1, $Channel_1$)	1
...	...	...

A vector of the distinct attribute value combination is called an element in this paper, denoted as $e=(p,i,d,c)$, where ($p\in\mathrm{P}$ or $p=*$), ($i\in\mathrm{I}$ or $i=*$), ($d\in\mathrm{D}$ or $d=*$), and ($c\in\mathrm{C}$ or $c=*$), $*$ is the wildcard. When $e=(p,i,d,c)$, $(p\neq *, i\neq *, d\neq *, c\neq *)$, we count the number of the PV records according to an element e in every time scale (e.g., the scale is each minute in this paper), and call this number PV value of the element, denoted by $v(e)$, i.e., $v(e) = \#\ records\ for\ e\ at\ a\ specific\ time\ scale$. Table III shows the PV values corresponding to the PV records in Table II.

The collection of all these most fine-grained elements, like the ones in Table III, are denoted by LEAF= $\{e|e = (p,i,d,c), p \neq *, i \neq *, d \neq * c \neq *\}$. The other elements, when one or more attribute value is $*$, can all be summed up based on the elements in $LEAF$. For instance, for the three elements at 10:00 (from 10:00:00 to 10:00:59) as in Table III, we can obtain the values of more coarse-grained elements, e.g.,

$$v(Beijing, Mobile, DC_1, *) = 2 + 1 = 3,$$
$$v(Beijing, *, *, *) = 2 + 1 + 3 = 6.$$

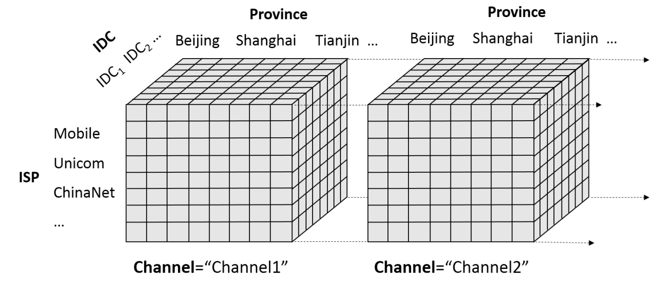


Fig. 1: A PV system of a 4-d data cube, represented as a series of 3-d data cubes

Based on the different degree of aggregation, we categorize the elements into different sets, and each set corresponds to a cuboid. A cuboid is a sub-cube of a data cube which is a data structure that allows data to be modeled and viewed in multiple dimensions [3], e.g., the elements of $LEAF$ constitute a 4-d data cube, as shown in Fig. 1. The cuboid is denoted as B_i (i can be an arbitrary combination of P, I, D and C), e.g., B_P is a 1-d cuboid and $B_{P,I,D}$ is a 3-d cuboid. The element set of a cuboid B_i is denoted as $E(Bi)$, e.g., $E(B_P) = \{e|e = (p,*,*,*), p \neq *\}$, $E(B_{P,I,D}) = \{e|e = (p,i,d,*), p \neq *, i \neq *, d \neq *\}$, $LEAF = E(B_{P,I,D,C})$.

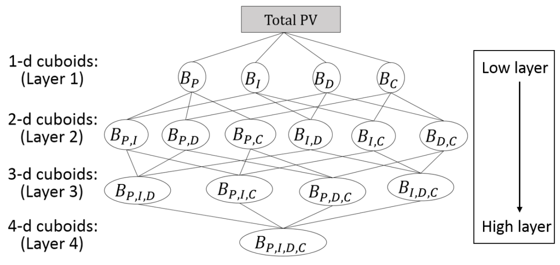


Fig. 2: Cuboids in a 4-d data

Moreover, we structure the cuboids and label layer IDs for them, as shown in Fig. 2. In addition, we say B_P or B_I is a father cuboid of $B_{P,I}$, and $B_{P,I}$ is a child cuboid of B_P or B_I. Accordingly, the elements of the cuboids, such as $(p,*,*,*)(\in E(B_P))$ and $(p,i,*,*)(\in E(B_{P,I}))$, also have the father-and-child relationships.

We denote $e' = (p',i',d',c')$ as a descendant of $e = (p,i,d,c)$ iff $(e \neq e')$ and $(p' = p$ or $p = *)$ and $(i' = i$ or $i = *)$ and $(d' = d$ or $d = *)$ and $(c' = c$ or $c = *)$. $Desc(e) = \{e'|e'$ is a descendant of $e\}$. $Desc'(e) = \{e'|e' = (p',i',d',c') \in LEAF, e' \in Desc(e)\}$. If $e \in LEAF$, then PV value $v(e)$ is directly measured. Otherwise,

$$v(e) = \sum_{e' \in Desc'(e)} v(e') \qquad (1)$$

e.g.,

$$v(Beijing,*,*,*) = \sum_{j,k,h} v(Beijing, i_j, d_k, c_h), \qquad (2)$$

$$Total\ PV = v(*,*,*,*) = \sum_{i,j,k,h} v(p_i, i_j, d_k, c_h). \qquad (3)$$

B. Problem Statement

The additive KPI (with multi-dimensional attributes) anomaly localization problem is to identify the cuboid and its elements that most potentially have caused the anomalous change of the total KPI value.

TABLE IV: A simple PV structure

$v(p,i)$		Province(p)			
		Beijing	Shanghai	Guangdong	*
ISP (i)	Mobile	20	15	10	45
	Unicom	10	25	20	55
	*	30	40	30	100(Total)

TABLE V: Example for problem statement

$f(p,i) \to v(p,i)$		Province(p)			
		Beijing	Shanghai	Guangdong	*
ISP (i)	Mobile	20→14	15→9	10→10	45→33
	Unicom	10→7	25→15	20→20	55→42
	*	30→21	40→24	30→30	100→75

To clarify the problem, we take a simple example in Table IV and Table V. Table IV shows a 2-d attributes PV structure. There exist two 1-d cuboids, B_P and B_I, and one 2-d cuboid $B_{P,I}$. Each cuboid contains a set of elements, i.e., $E(B_P) = \{(Beijing,*), (Shanghai,*), (Guangdong,*)\}$, $E(B_I) = \{(*,Mobile), (*,Unicom)\}$, $LEAF = E(B_{P,I}) = \{(Beijing,Mobile), (Shanghai,Mobile), (Guangdong,Mobile), (Beijing,Unicom), (Shanghai,Unicom), (Guangdong,Unicom)\}$. $v(p,i)$ are shown in the cells of the table, e.g., $v(Beijing,Mobile) = 20$, $v(Beijing,*) = 30$.

When the total PV is anomalous, the PV changes are shown in Table V. In each cell, the first number is the forecast PV value $f(p,i)$, and the second is the actual PV value $v(p,i)$ (how to detect the total PV and calculate the elements' forecast values will be introduced in §IV-A). The forecast value of total PV is 100, while its actual PV value is only 75 (the bottom right corner of Table V). Hence an alert is generated because of the anomalous change of the total PV ($v(*,*)$=75 is much smaller than $f(*,*)$=100) that triggers anomaly localization.

Regarding the three cuboids, B_P, B_I and $B_{P,I}$, they can express the PV KPI from different perspectives. When the total PV is changed anomalously, each of these three cuboids is impacted. As shown in Table V, there are some anomaly elements in each cuboid (the shaded cells). In reality, operators need to determine which cuboid and which elements of this cuboid are the most potential root cause for this anomaly. Then they can initiate the attempt to fix the anomaly and mitigate loss. Therefore, the problem of anomaly localization for additive KPIs can be restated as follows:

Effectively and efficiently identify the most potential root cause, i.e., a subset of elements of one specific cuboid B_i, for a total KPI value anomaly. The root cause set $RSet \subseteq E(B_i)$.

Note that this definition allows the multiple elements within the same cuboid as the root cause set. For instance, the root cause set of the example in Table V is $RSet = \{(Beijing,*), (Shanghai,*)\}$. However, this definition excludes the cases where there are simultaneous root causes in multiple cuboids, which is extremely rare in reality. Also note that we only deal with the case where total KPI value is anomalous.

C. Challenges

There are mainly two challenges for our anomaly localization problem.

How to measure the potential of an element set to be the root cause is not easy. To localize the most potential set to be the root cause, we have to define a value function to measure the potential of each set. However, some intuitive metrics, e.g., change or change proportion, do not work well. We denote the change of the PV value of an element by $h(e)$, and it can be calculated by $h(e) = f(e) - v(e)$. The change of a set S of elements is $h(S) = \sum h(e), e \in S$. Now consider the example in Table V. The total PV is changed by $h(total) = f(total) - v(total) = 100 - 75 = 25$. The shaded cells are the changed elements. Consider the two sets, $S1 = \{(Beijing,*), (Shanghai,*)\}$ and $S2 = \{(*,Mobile), (*,Unicom)\}$, in cuboids B_P and B_I respectively. We find that the changes of the two sets are equal, i.e., $h(S1) = h(S2) = 25$, and this change can cover the total PV change 100%. So the change or change proportion (change proportion, denoted as r, i.e., $r(e) = \frac{h(e)}{h(total)}$, here means 100%) cannot distinguish which set is more potential to be the root cause, but in reality S1 is the true root cause that should be more "potential" than S2. Hence, it is not easy to find an appropriate approach to measure the potential of an element set. For this reason, we need to define a potential score (ps) that can measure the potential degree of a set, which will be elaborated in Section III and IV.

There are too many sets that need be compared. As mentioned above, we will define a potential score to measure how potential an element set is to be the root cause. We aim to find the subset of each cuboid with the largest potential score. In addition, we can tell that in advance, the potential score of elements are non-additive, i.e., $ps(\{e_1, e_2\}) \neq ps(\{e_1\}) + ps(\{e_2\})$. Thus we need to calculate and compare all subsets for each cuboid in principle. That is, for each cuboid, we need to list all the subsets exhaustively and calculate their potential scores. Here "to list all the subsets exhaustively" is rather complicated. E.g., in Table V the cuboid B_I has two elements, i.e., $E(B_I) = \{(*,Mobile)\}, \{(*,Unicom)\}$, so three sets can be listed, $\{(*,Mobile)\}, \{(*,Unicom)\}$ and $\{(*,Mobile),\ (*,Unicom)\}$. Actually, if a cuboid has n elements, the number of all possible subsets will be $2^n - 1$, except $\varnothing$. In practice, n can be very large, even more than tens of thousands. For instance, let n be 100, then the set number will be $2^{100} - 1$. Thus it is too large of a set space to be able to search and calculate each potential score.

III. Core idea and overview

To tackle the two challenges mentioned in Section II-C, we need to do: 1) Propose an function to measure the potential of element sets to be the root cause; 2) Find an efficient method to search all possible sets (to be the root cause). In this paper, we propose Potential Score as the metrical function, and apply Monte Carlo Tree Search (MCTS) algorithm and hierarchical pruning strategy to overcome the huge search space problem. We briefly introduce them next.

A. Potential Score for measuring the potential of sets

In our anomaly localization problem, a metric that can be used to "globally" compare the root cause "potential" of different element sets. However, as shown in the first challenge, such a metric is not easy to develop and naive metrics do not work.

Our idea for this Potential Score is based on the following intuition: when the KPI value at a root cause element changes, all its descendant LEAF elements' KPI values also change accordingly. Thus the "potential score" of a candidate root cause element is then to gauge the difference between the expected and actual changes of this element's descendant LEAF elements. See more details in §IV-B2. In addition, MCTS needs Potential Score as a value function to guide the searching.

B. MCTS and Hierarchical Pruning for efficiently searching

The huge search space in our problem requires an effective and efficient searching algorithm. Our intuition in this paper is to adopt some advanced algorithm that are known to be good at searching in huge space, instead of developing organic heuristic algorithm as previous works did with their simpler anomaly localization within much smaller search space [1], [2]. Inspired by AlphaGo's successful adoption of MCTS in Go game [4], [5], the core idea of this paper is thus to adopt MCTS as the base algorithm in our anomaly localization solution. However, there are still a remaining challenge in adopting MCTS and we now summarize our core ideas to address them.

From Fig. 2 we can see that as we go from lower layer to higher layer, the number of elements n in a cuboid becomes larger and larger. For example, there are 36 elements in B_P, 36*10 in $B_{P,I}$, and 36*10*6*10 for $B_{P,I,D,C}$. Recall that the root cause set is one of the $(2^n - 1)$ subsets of a cuboid. Searching such a huge space is no easy task even for MCTS.

To future reduce the search space, HotSpot applies a hierarchical pruning strategy. The basic idea is that, after searching lower layers, HotSpot prunes some elements (in higher layers) that is unlikely to be root cause elements. The intuition is that if a father element has a very low potential score, each of the children elements is unlikely to be a root cause element , and, thus can be pruned. This approach in spirit is very similar to the Apriori Principle in Association Rule Mining [3]. We call our pruning approach hierarchical pruning because its pruning policy utilizes layer hierarchy information. See more details in §IV-D.

C. Overall Approach

The core ideas of HotSpot are summarized as follows. We consider this anomaly localization as a search problem with a huge space; Adopt MCTS as our base searching algorithm; Propose a potential score metric (with physical significance in anomaly localization) as the potential measure for each set and the value function in MCTS; Apply a hierarchical pruning approach (similar to the Apriori Principle in spirit) to future reduce the search space. Searching starts from layer 1 and is done layer by layer, and MCTS is applied within each cuboid, as shown in Fig. 3.

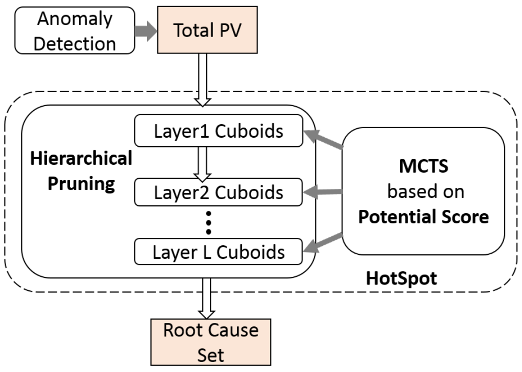


Fig. 3: The overview of HotSpot

IV. Design of HotSpot

This section presents the detailed design of HotSpot. HotSpot searches the sets of cuboids layer by layer, i.e., from layer 1 to layer L (L is the number of layers). For each cuboid of a given layer, HotSpot applies MCTS to find its subset with the largest potential score (ps), which is called best set (abbreviated Bset) of this cuboid. When going from a layer to the next layer, hierarchical pruning is used. We repeat this process until layer L is searched, or the root cause set RSet ($ps(RSet) > PT$) is obtained, where PT (ps Threshold) is a threshold that we think it is large enough to be regarded as the root cause set when a set with a $ps > PT$. The final output RSet is the $BSet$ with the largest ps among all $BSets$ generated by the algorithm. Next we describe a method to detect the total KPI and forecast the elements in this section. Then we present each component of HotSpot, i.e., potential score, MCTS and hierarchical pruning.

A. Anomaly Detection and Forecast

HotSpot needs an anomaly detection algorithm 1) to detect anomalies in the total KPI, and 2) to calculate the forecast values of other elements (see §IV-B for more details).

We adopt a statistical algorithm which has been widely used in the industry for anomaly detection [6] on the total KPI. The mean (μ) and the standard deviation (σ) are calculated for each time interval (in our case, each minute) of the week, where μ is regarded as the forecast value. The thresholds (T_l and T_u stand for the lower and upper thresholds, respectively) are defined as follows:

$$T_l = \mu - c \times \sigma,\ T_u = \mu + c \times \sigma \tag{4}$$

where c is a parameter that determines the degree of the upper and lower thresholds (usually set as 2.0) [6]. Note that the thresholds are updated periodically. An anomaly is detected if the actual value is beyond the thresholds. This algorithm is suitable in our scenario because 1) it fits very well with additive KPI data for most of additive KPI data is periodic, and 2) it is computationally efficient.

B. Potential Score

TABLE VI: An anomalous element $(Beijing, *)$ for PV

$f(p,i) \rightarrow v(p,i)$		Province(p)			
		Beijing	Shanghai	Guangdong	*
ISP (i)	Mobile	20→8	15→15	10→10	45→33
	Unicom	10→4	25→25	20→20	55→49
	*	30→12	40→40	30→30	100→82

1) Ripple effect: We use a new anomaly case in Table VI to illustrate how the KPI change at the root cause element is propagated to other elements according to the "ripple effect" that we summarize. The PV of $(Beijing, *)$ is decreased from 30 ($f(Beijing, *)$) to 12 ($v(Beijing, *)$), and $(Beijing, *)$ is the only root cause element in this case. Since $v(Beijing, *)$ is aggregated by its descendant elements, $v(Beijing,\ Mobile)$ and $v(Beijing,\ Unicom)$, they must have changed correspondingly. Note the change value of them, $h(Beijing, *) = 18$, $h(Beijing,\ Mobile) = 12$ and $h(Beijing,\ Unicom) = 6$. We can get that the actual value $v(Beijing, Mobile) = 8$ equals to its proportional share according to the formula $f(Beijing, Mobile) - h(Beijing, *) \times \dfrac{f(Beijing, Mobile)}{f(Beijing, *)} = 20 - 18 * \dfrac{20}{30}$. In addition, $h(Beijing, Mobile)$ in turn contributes to the change in $v(*,\ Mobile)$.

The above example illustrates how a root cause element affects its descendant elements (in $LEAF$) and other elements which share a common descendant element with it. Generally, when the value of a root cause element increases or decreases, it obeys the ripple effect property as follows:

Let x denote an element that is not in LEAF, i.e., $x \notin LEAF$. Let x'_i denote the descendant elements of x in $LEAF$, i.e., $x'_i \in Desc'(x)$. When the PV value of x changes by $h(x)$, i.e., $h(x) = f(x) - v(x)$, x'_i will get its share of $h(x)$ according to the proportions of their forecast values, i.e.,

$$v(x'_i) = f(x'_i) - h(x) \times \frac{f(x'_i)}{f(x)},\ (f(x) \neq 0). \tag{5}$$

Then all other elements e who are ancestors of x'_i are updated using Eq. 1.

The above ripple effect describes the situation that the root cause contains just one element. When it comes to a set (two or more elements), we can reuse the property for each element.

2) Potential Score: The ripple effect reveals how a root cause set affects many other elements' values. Therefore, to measure the potential of a set to be the root cause, we propose to 1) assume that the set S is the root cause, 2) deduce new PV values of the descendant elements in LEAF based on the ripple effect, and 3) compare all the actual PV values with the newly deduced PV values of LEAF elements. The closer the two kinds of values are, the more potential the set has to be the root cause set.

Let $y_1, y_2, y_3, ..., y_n$ express all the elements of $LEAF$. We denote the newly deduced PV values of an assumed root cause set S with $a(y_i)$. We compute $a(y_i)$ in two situations: a) $S \nsubseteq LEAF$: $if\ \ y_i \notin Desc'(S),\ a(y_i) = f(y_i)$, else $a(y_i)$ is computed using Eq. (5); b) $S \subseteq LEAF$: $if\ \ y_i \notin S,\ a(y_i) = f(y_i)$, else $a(y_i) = v(y_i)$. Let $\vec{a}$ be the vector of $a(y_i)$, i.e., $\vec{a} = [a(y_1), a(y_2), a(y_3), ..., a(y_n)]$. Similarly, let $\vec{v} = [v(y_1), v(y_2), v(y_3), ..., v(y_n)]$, $\vec{f} = [f(y_1), f(y_2), f(y_3), ..., f(y_n)]$.

Then we define the Potential Score (ps) of a set S:

$$Potential\ Score = max(1 - \frac{d(\vec{v}, \vec{a})}{d(\vec{v}, \vec{f})}, 0) \tag{6}$$

where $d(\vec{u}, \vec{w})$ represents the distance of the vectors $\vec{u}$ and $\vec{w}$. Here we adopt the Euclidean distance:

$$d(\vec{u}, \vec{w}) = \sqrt{\sum_i (u_i - w_i)^2}. \tag{7}$$

The potential score of a set ranges from 0 to 1, i.e., [0,1]. If a set has a higher score, it will be considered to have higher potential to be the root cause.

Above definition of potential score is "global" in the sense that any two element sets can compare their potential scores to see which one has more potential. This serves a good value function necessary in MCTS.

When two element sets have the same potential score, we follow a "succinctness" principle. i.e., the one with less number of element wins, either following the Occam's razor principle [1] or because the elements of one set are collectively the ancestors (preferred as root cause) of those in the other.

3) An illustrating example: Now we illustrate how to find the root cause based on potential score for the case in Table V. The cuboids are B_P, B_I and $B_{P,I}$. The best set of each cuboid (the subset with the largest potential score of this cuboid) will be found at first. Next we choose the root cause set by comparing the best sets. $\vec{y}$ is denoted in this order $[(Beijing, Mobile), (Shanghai, Mobile), (Guangdong, Mobile), (Beijing, Unicom), (Shanghai, Unicom), (Guangdong, Unicom)]$. Then $\vec{f} = (20, 15, 10, 10, 25, 20)$, $\vec{v} = (14, 9, 10, 7, 15, 20)$. For the cuboid B_P, it contains three elements $(Beijing, *)$, $(Shanghai, *)$ and $(Guangdong, *)$, so all the subsets are $S_{p1} = \{(Beijing, *)\}$, $S_{p2} = \{(Shanghai, *)\}$, $S_{p3} = \{(Guangdong, *)\}$, $S_{p4} = \{(Beijing, *), (Shanghai, *)\}$, $S_{p5} = \{(Beijing, *), (Guangdong, *)\}$, $S_{p6} = \{(Shanghai, *), (Guangdong, *)\}$ and $S_{p7} = \{(Beijing, *), (Shanghai, *), (Guangdong, *)\}$. Take the set S_{p1} as an example, using Eq. (5), we can get the deduced PV values, $\vec{a}(S_{p1}) = (14, 15, 10, 7, 25, 20)$. Then the ps can be obtained, $ps(S_{p1}) = 0.13$. Actually, we can find that both S_{p4} and S_{p7} have the largest ps, $ps(S_{p4}) = ps(S_{p7}) = 1$. Considering the goal of succinctness, S_{p4} is the best set in B_P. Similarly, we can obtain two other best sets for B_I and $B_{P,I}$, $S_{i3} = \{(*, Mobile), (*, Unicom)\}$ with $ps(S_{i3}) = 0.47$ $(\vec{a}(S_{i3}) = (14.67, 11, 7.33, 7.64, 19.09, 15.27))$ and $S_{pi1} = \{(Beijing, Mobile), (Beijing, Unicom), (Shanghai, Mobile), (Shanghai, Unicom)\}$ with $ps(S_{pi1}) = 1$ $(\vec{a}(S_{pi1}) = (14, 9, 10, 7, 15, 20))$. Comparing the three best sets, S_{p4} is the result set with the largest ps and the most succinctness.

The example above illustrates our core idea of using potential score to identify the root cause set. Actually, the elements are too many so that the number of possible sets is extremely massive, especially in the cuboids of higher layers. To handle this problem, we apply MCTS algorithm and hierarchical pruning strategy which will be introduced next. At the same time, using the two methods can help in finding the succinct result.

C. MCTS Algorithm

For a given cuboid B, we want to obtain the best set (the subset with the largest potential score of this cuboid). Suppose there are n elements in $E(B)$. The search space within B for the root cause set is $2^n - 1$, which apparently can be very large for a large n. HotSpot adopts MCTS mainly to tackle this challenge of search space explosion.

MCTS is a heuristic method for searching optimal decisions in a given domain by taking random samples in the decision space and building a search tree according to the results from existing random examples. At the very high-level, MCTS tries to balance the exploitation along those promising branches and the exploration along those unexplored branches. It has been widely used in the Artificial Intelligence (AI) field for domains that can be represented as trees of sequential decisions, particularly games and planning problems [4], such as AlphaGo [5].

In MCTS, each node represents a state s (the *root* can be regarded as $\varnothing$). An action space $A(s)$ contains all the legal actions that can be taken at s. The algorithm can move from one state s to another by taking a legal action, named $a \in A(s)$, via the edge (s,a). There can be variables associated with an edge, used by the algorithm to indicate the "value" of taking action a at state s.

We adopt MCTS to our anomaly localization problem in a cuboid as follows. We first calculate $ps(e)$ for each e in this cuboid, and rank all e according to $ps(e)$. Each state s corresponds to the candidate root cause set $S(s)$ that is currently being explored. $N(s)$ is the number of times s has been visited. We setup three variables for each edge (s,a). $N(s,a)$ is visit count, i.e., the number of times that edge (s,a) has been visited. $ps(S(s))$ is the potential score of set $S(s)$. Suppose s transitions to s' following (s,a). Then edge (s,a)'s action value $Q(s,a) = \max_{u \in \{s'\} \cup descendent(s')} ps(S(u))$, which equals the maximum potential score of s' and its descendant nodes in the tree. $Q(s,a)$ is initialized to be $ps(S(s))$ for each s.

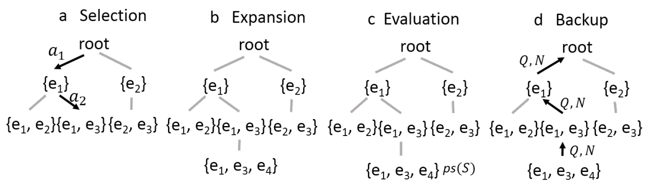


Fig. 4: Monte Carlo Tree Search in HotSpot

Now we illustrate the four steps of a MCTS iteration in our anomaly localization. Suppose that at the beginning of the current iteration, the state tree is as shown in Fig. 4(a).

a) Selection. The goal of this step is to select a node from the current state tree to be expanded. Each time when this step is executed, the tree traversal always starts with the root state. Assume that we have advanced to the current state s in this selection step. If all the actions in $A(s)$ have been visited in previous iterations, then an action a is selected from the set of available actions $A(s)$ by using the Upper Confidence thresholds (UCB) algorithm [7], shown as Eq. 8.

$$a = \underset{a \in A(s)}{\arg\max} \{Q(s,a) + C\sqrt{\frac{\ln N(s)}{N(s,a)}}\}. \tag{8}$$

$Q(s,a)$ is the value of taking the move a. The higher the value of $Q(s,a)$, the larger the chance of move a is selected in this selection step, which is the exploitation mechanism in MCTS. The second part of the equation is just the standard UCB mechanism for exploration. The balance between exploitation and exploration can be changed by modifying C. A commonly used value of C is $\sqrt{2}$ [8], which we choose in this paper, or it can be chosen empirically in practice.

In case there is an action $a \in A(s)$ that has not been explored at all, Eq. 8 cannot be applied since $N(s,a) = 0$. Instead, we assign a probability of taking unvisited actions to be $R = (1 - Q(s, a_{max}))$, where $a_{max} = \arg\max_{a \in A(s) \cap N(s,a)=0} Q(s,a)$.

The selection step starts at the root of the tree, and stops when a leaf state is chosen according to Eq. 8 or an unvisited action is selected. E.g., in Fig. 4(a) along with the bold edges, the selection step stops when the leaf state $\{e_1, e_3\}$ is selected.

b) Expansion. After a state s is selected in the selection step, we then expand the Monte Carlo Tree by adding a new node s', where $S(s') = S(s) \cup \{e^*\}$ and $e^* = \arg\max_{e \in \{e_1, e_2, \ldots, e_n\} - S(s)} ps(e)$. We choose e^* to have the largest $ps(S)$ value of the remaining elements rather than choosing e^* randomly. For example, in Fig. 4(b), s (where $S(s) = \{e_1, e_3\}$) is selected, and then $e^* = e_4$ is added to get s' where $S(s') = \{e_1, e_3, e_4\}$.

c) Evaluation. To initialize the new node after expansion (e.g., $\{e_1, e_3, e_4\}$ in Fig. 4(c)), we calculate its ps, Q and N.

d) Backup. Action values Q and visit count N on all nodes along path from s' to the root are updated, as illustrated by the bold arrows in Fig. 4(d). Recall the definition of Q, along the path, we update the Q of a father only when the child's Q is greater than the father's.

Localizing the root cause set in a cuboid. We apply MTCS in each cuboid, for which we iteratively perform the above four steps until at least one of the following three conditions occur:

1) A best set is found, i.e., $BSet = S$ if $ps(S) \geqslant PT$;
2) All the available nodes of the set are expanded;
3) The iteration time is greater than a maximum number M, which is configured empirically.

Under both the second and third terminating conditions, if we have not obtained a set whose ps is greater than PT, we will return the $BSet$ with the greatest ps as the $RSet$.

D. Hierarchical Pruning

In order to further reduce the search space for the cuboids in higher layers, HotSpot applies a hierarchical pruning strategy. The basic idea is that, HotSpot searches the cuboids layer by layer, i.e., from layer 1 to layer L. After searching a lower layer, it prunes some elements in the higher layers that is unlikely to be root cause elements.

For each cuboid B of layer l $(1 \leqslant l < L)$, we can obtain the best sets (the subset with the largest potential score of this cuboid) $BSet_{l,B}$ using the MCTS algorithm. Our intuition is as follows. If an element $(p_1, i_1, *, *)$ in layer $l+1$ has a high potential score, its father elements $(p_1, *, *, *)$ and $(*, i_1, *, *)$ in layer l will also have a relatively high potential score. Therefore, if a father element has a very low potential score, each of the children elements is unlikely to be a root cause element, although there can be rare cases where a children element a does have a potential score higher than its father element's but some other children element b 's PV changes cancel off a's effect on the father's potential score. As a result, if an element in layer l is not in $BSet_{l,B}$, HotSpot chooses to prune all its children elements. This approach in spirit is very similar to the Apriori Principle in Association Rule Mining [3]. We call our pruning approach hierarchical pruning because its pruning policy utilizes layer hierarchy information.

TABLE VII: Example for hierarchical pruning

$f(p,i) \rightarrow v(p,i)$		Province(p)			
		Fujian	Jiangsu	Zhejiang	*
ISP (i)	Mobile	20→5	15→15	10→10	45→30
	Unicom	10→10	25→13	20→20	55→43
	*	30→15	40→28	30→30	100→73

We take an example in Table VII and illustrate our hierarchical pruning approach in Fig. 5. Suppose we are in layer 1, and the best sets obtained using MCTS are $BSet_{1,B_P} = \{(Fujian,*),(Jiangsu,*)\}$ with $ps(BSet_{1,B_P}) = 0.50$, and $BSet_{1,B_I} = \{(*,Mobile), (*,Unicom)\}$ with $ps(BSet_{1,B_I}) = 0.32$. When searching cuboids in layer 2, we prune the elements $(Zhejiang, Unicom)$ and $(Zhejiang,Unicom)$ because their father element $(Zhejiang,*)$ is not in the $BSet$s of layer 1. Therefore, we only need to search the remaining four elements for $B_{P,I}$. This way, the number of potential sets will be reduced from 63 to 15 (2^6-1 to 2^4-1). Then using MCTS again in layer 2, we obtain the $RSet = BSet_{2,B_{P,I}} = \{$(Fujian, Mobile), (Jiangsu, Unicom)$\}$, where $ps(BSet_{2,B_{P,I}}) = 1$.

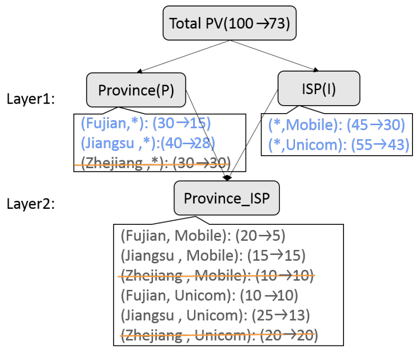

Total PV(100→73)
Layer1:
Province(P)
ISP(I)
(Fujian,*): (30→15)
(Jiangsu ,*):(40→28)
~~(Zhejiang ,*): (30→30)~~
(*,Mobile): (45→30)
(*,Unicom): (55→43)
Layer2:
Province_ISP
(Fujian, Mobile): (20→5)
(Jiangsu , Mobile): (15→15)
~~(Zhejiang , Mobile): (10→10)~~
(Fujian, Unicom): (10→10)
(Jiangsu , Unicom): (25→13)
~~(Zhejiang , Unicom): (20→20)~~

Fig. 5: Hierarchical pruning for example in Table VII.

E. The Overall Algorithm

We now summarize our overall HotSpot algorithm, whose pseudo code is shown in Algorithm 1. HotSpot takes the PV values of elements, a potential threshold PT and a maximum iteration number M as inputs. It starts with layer 1. For each cuboid of a given layer, HotSpot applies MCTS to find its best set. When going from a layer to the next layer, hierarchical pruning is used. We repeat this process until layer L is searched, or the root cause set $RSet$ ($ps(RSet) > PT$) is obtained. The final output $RSet$ is the the $BSet$ with the greatest ps among all $BSet$ generated by the algorithm.

V. Evaluation

In this section, we evaluate the performance of HotSpot using comparison experiments driven by synthetically injected anomalies on top of real-world PV data.

Algorithm 1 HotSpot

```
Input:
  All the PV values of elements
  PT: Potential Threshold
  M: Maximum number of Iteration
Output:
  RSet: Root cause set
  procedure Anomaly localization:
    The total PV is found anomalous.
    // The strategy of  hierarchical pruning
    for layer $l$ in [1,L] do // L is maximum ID of Layer
      // Parallel Execution in each cuboid
      for each cuboid $B_j$ of current layer $l$ do
        Calculate Potential Scores $ps(e_k)$ of each element $e_k$
        Sort $e_k$ in a descending order of $ps(e_k)$
        // $i$ is the number of iteration now, and be initialed 0
        $i = 0$
        // E is the list of sorted elements
        // $BSet_{l,j}$ is the best set of $B_j$ in layer $l$
        To find $BSet_{l,j}$ of E by MCTS:
        while True do
          Choose a $set$ use UCB algorithm
          if $i \geqslant M$ then
            break
          end if
          if $ps(set) \geqslant PT$ then
            $RSet = set$
            return (RSet)
          end if
          $i = i + 1$
        end while
        Obtain $BSet_{l,j}$
        Prune $e_c$ in layer $l+1$ whose father $e_f$ are not
        in $BSet_{l,j}$
        if All the $e_c$ in layer $l+1$ are pruned then
          break
        end if
      end for
    end for
    // Choose $RSet$ form $BSet_{l,j}$ with the largest $ps$
    $ps(RSet) = Max\{ps(BSet_{l,j})\}$
    return (RSet)
  end procedure
```

A. Dataset

Now we introduce the dataset in our evaluation. We collect the PV records from a top global search engine for nine weeks. The data has a periodicity of one week. The last week data is used for injecting anomalies and testing, and the former eight weeks data is used as historical data for calculating the mean (μ) and the standard deviation (σ) (mentioned in §IV-A). The number of PV records is about 10.8 billion everyday. As aforementioned, a record can be "10:00:01; $Beijing$, $Mobile$, DC_1, $Channel_1$", and the granularity of a timestamp is second. Each record has four attributes, which are P, I, D and Channel C. Recall that we can calculate the PV values of LEAF elements by counting the corresponding records for each time interval (the interval is one minute here). Then the PV values of each cuboid's elements can be aggregated using Eq. 1. Table VIII shows all the 15 cuboids in this dataset and the number of elements in each cuboid (in the parenthesis).

B. Injecting Synthetic Anomalies

To thoroughly evaluate HotSpot and compare it with other approaches, ideally we would like to use the set of

TABLE VIII: The cuboids across four layers.

Layer ID	15 cuboids. The no. of elements in a cuboid is shown in parenthesis.					
Layer 1	B_P (36)	$B_I(10)$		$B_D(6)$	$B_C(10)$	
Layer 2	$B_{P,I}(360)$	$B_{P,D}(216)$	$B_{P,C}(360)$	$B_{I,D}(60)$	$B_{I,C}(100)$	$B_{D,C}(60)$
Layer 3	$B_{P,I,D}(2160)$	$B_{P,I,C}(3600)$		$B_{P,D,C}(2160)$	$B_{I,D,C}(600)$	
Layer 4	$B_{P,I,D,C}$ (21600)					
	Total number of elements: 31338					

anomalies that cover the entire parameter space. For this purpose, the anomalies in the real data set is often too few and the root causes of them are often not comprehensive (coverage of various layer IDs and the number of elements in the root cause set). Instead, similar to many previous works (e.g., [9]), we smooth the data to get ride of the major fluctuations in the existing data of the elements, and then inject synthetic anomalies into the last week's data. There are four steps in anomaly injection.

First, for each LEAF element, we smooth the data using moving average method Second, we add Gaussian noises onto the smoothed LEAF elements' data using the following equation $v^* = v + \alpha * N(0, \sigma^2)$, where σ is the standard deviation value and α is chosen to make the anomalies obvious. All the other elements can then be calculated using Eq. 1. Fig. 6 shows the original and the smoothed values of an example element. Third, for an injected anomaly at a specific element (at the last week), we use the ripple effect to "distribute" the difference between the forecast value and the anomalous value to its descendant LEAF elements, with Gaussian noises added. The anomalies injected are spikes of multiple minutes (considering the monitoring interval is minute), during which anomalies are detected and the anomaly localization are conducted. Fourth, we use Eq. 1 to aggregate the LEAF elements to obtain all other elements' PV values. Fig. 7 shows the values after injecting anomalies and noises.

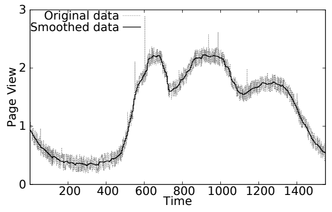


Fig. 6: The original and smoothed values of an element

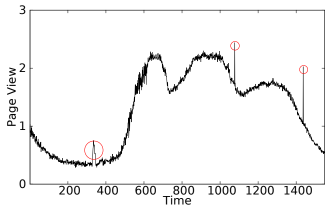


Fig. 7: Smoothed values with synthetic anomalies and noises

C. Injecting Root Cause Cases

A root cause case, i.e., a root cause set, can be in any layer, and it usually contains a certain number of elements. Obviously, the number of elements and the position (i.e., in which layer) of the anomaly case can significantly impact the computational cost and the accuracy of the anomaly localization methods. As aforementioned, there are four layers in our dataset. While HotSpot can handle root cause set of n elements, in the following evaluation we limit n to be up to 5, which is sufficient to show the improvement over previous work. Hence there are 20 different types of cases, i.e., type 1: "layer 1 and 1 element in each case", type 2: "layer 2 and 1 element in each case", ..., type 20: "layer 4 and 5 elements in each case". Then, how many cases should be assigned to each type? Obviously, the actual case distribution can vary for different application scenarios. In our scenario, the number of actual cases is too few to help answer this question. We opt to equally inject a number of cases for the 20 types (400 cases for each type), thus the results here show HotSpot's performance under various conditions rather than the performance under the realistic anomaly distribution (which unfortunately is hard to get).

D. Metrics and Experimental Hardware

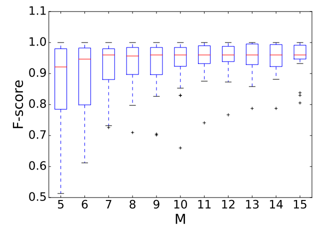
We use Precision, Recall and F-score metrics to evaluate the accuracy of HotSpot. The F-score is defined as $F-score = \frac{2 * Precision * Recall}{Precision + Recall}$, where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$. TP (true positive) is the number of root cause elements correctly reported. FP (false positive) is the number of the root cause elements wrongly reported. FN (false negative) is the number of anomaly elements that is not reported. The higher the metric is (Precision, Recall or F-score), the better the approach performs.

The experiments were run on a server with (24-core Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz with 64GB RAM).

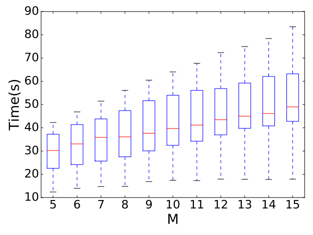
E. Parameter Determination

There are two parameters that need to be pre-configured in HotSpot, i.e., the potential threshold PT and the maximum iteration number M of MCTS. PT is a stopping condition of the HotSpot procedure. It can be tuned by operators according to the specific requirement in practice. Specifically, the closer a PT value is to 1, the more precise the algorithm is. The operators we worked with would like to have very accurate results if possible, we thus set PT as 0.99, which means that if we find a set with $ps > 0.99$, we will stop searching and regard it as the root cause.

The maximum iteration number M of MCTS can greatly affect the effectiveness and efficiency of HotSpot. To improve computational efficiency, MCTS conducts a technically local search instead of traversing the entire search space. Qualitatively speaking, the more times the MCTS iterates, the more accurate the result will be, but the cost time will be longer. However, M cannot be set by operators due to the lack of physical significance to them, and we run HotSpot using various M values, i.e., from 5 to 15, and empirically select a rather reasonable M value by balancing effectiveness and efficiency.



(a) The box-plots of F-scores of various M values



(b) The box-plots of running time of various M values

Fig. 8: The performance of HotSpot under various maximum iteration numbers (M). The bottom and top of the box are the first and third quartiles ($Q1$ and $Q3$), and the band inside the box is the median ($Q2$). The lower whisker is the minimum value within $Q1-1.5*IQR$, and the upper whisker is the maximum value within $Q3+1.5*IQR$, where $IQR=Q3-Q1$.

For each of the 20 case types, we injected 400 cases. M is ranged from 5 to 15. Hence for each M value, we can calculate the average F-score for each case type. In Fig. 8(a) and 8(b), we show the box-plots of F-scores and those of the running time for the 20 anomaly case types under different M values. Fig. 8(a) shows that the F-score of HotSpot increases with M. We observe that when $M>10$, the F-score becomes stable and the first quantile is larger than 90%, which empirically meet our demand. Fig. 8(b) shows that the running time linearly increases with M. When $M=10$, the third quantile is about 54s, which is acceptable by operators based on real-world investigation. Consequently, we set $M=10$ for HotSpot in the studied company.

F. The Effectiveness of HotSpot

To evaluate the accuracy of HotSpot, we injected anomaly cases using the methods in §V-C, we set $M=10$ as aforementioned. We compare it with two previously proposed approaches, i.e., Adtributor [1] and iDice [2].

Adtributor focuses on the revenue debugging problem, which is similar to HotSpot. However, it only deals with the root cause set in the layer 1 cuboids, while HotSpot takes all the cuboids (especially the multiple dimensional cuboids) into account.

iDice identifies the effective attribute combinations of an emerging issue for a large-scale software system. The multi-dimensional attribute space in iDice is very similar with that of HotSpot. In addition, an attribute combination is similar to an element in our system. However, iDice is tailored to the simpler cases where there are fewer “elements” in a “root cause set” in iDice (usually there are only one or two “elements” in a “root cause set”). Three parameters should be pre-configured in iDice. The default parameters in the original iDice paper [2] performed poorly in our experiments. As such, we swept iDice’s parameter space, and eventually settled with the combination of iDice’s parameters which achieved the best accuracy.

Fig. 9 shows the comparison of the F-scores of the three algorithms. Compared with iDice and Adtributor, HotSpot achieved higher F-scores across all the 20 types of cases (differentiated by layer ID and the number of elements in each case). The F-score of iDice decreased sharply as the number of elements increases. Although Adtributor achieved excellent accuracy in layer one anomaly cases, its accuracy dropped to zero when the cases were in higher layers. In contrast, HotSpot performed quite robust across different number of elements in each case, and different layers.

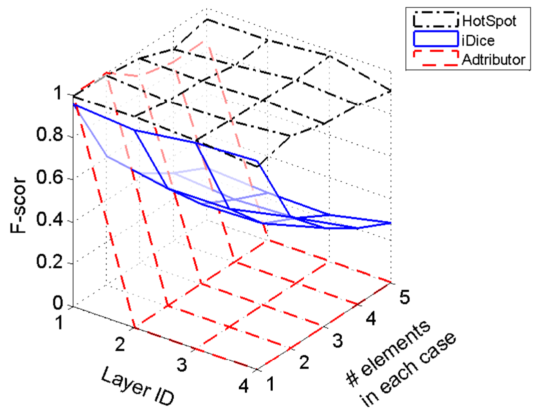


Fig. 9: The F-score comparison of the three algorithms

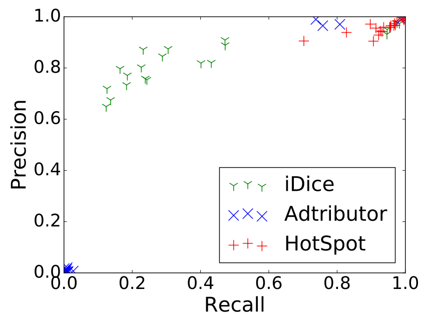


Fig. 10: The precision-recalls of three algorithms (Note that we add jitters to the overlap points for clearer display.)

The average precision and recall (over 400 cases) of each of 20 case types for three algorithms are shown in Table IX. Fig. 10 shows the distribution of the precision-recalls of the three algorithms across the 20 case types. In this figure, the precision-recall points of HotSpot are centralized in the upper right corner, demonstrating HotSpot’s robustness in accuracy. However, the precision-recall points of iDice in Fig. 10 are much more scattered than HotSpot, demonstrating that the accuracy of iDice are not robust against different types of anomaly cases.

TABLE IX: The comparison of three algorithms' precision and recall

Algorithms	Avg. over 400 runs	Single RC element				Two RC elements				Three RC elements				Four RC elements				Five RC elements			
		layer1	layer2	layer3	layer4	layer1	layer2	layer3	layer4	layer1	layer2	layer3	layer4	layer1	layer2	layer3	layer4	layer1	layer2	layer3	layer4
HotSpot	Precision	0.991	0.995	0.983	0.958	0.988	0.965	0.973	1	0.964	0.958	0.968	0.955	0.905	0.929	0.960	0.940	0.943	0.944	0.970	0.905
	Recall	0.995	0.995	0.983	0.958	1	0.961	0.973	1	0.968	0.957	0.968	0.916	0.908	0.923	0.937	0.831	0.928	0.925	0.895	0.704
iDice	Precision	0.955	0.928	0.966	0.985	0.91	0.82	0.82	0.888	0.84	0.75	0.758	0.87	0.8	0.735	0.77	0.867	0.72	0.67	0.65	0.803
	Recall	0.955	0.935	0.975	0.985	0.455	0.418	0.423	0.475	0.282	0.252	0.258	0.317	0.2	0.185	0.193	0.228	0.145	0.134	0.132	0.171
Adtributor	Precision	1	0	0	0	1	0	0	0	0.807	0	0	0	0.739	0	0	0	0.764	0	0	0
	Recall	1	0	0	0	1	0	0	0	0.881	0	0	0	0.844	0	0	0	0.849	0	0	0

Most of the precision-recall points of Adtributor are centralized in the bottom left corner except for the five precision-recall points of anomaly cases in layer one. In short, Fig. 9 and 10 both show that HotSpot is more accurate and robust than iDice and Adtributor.

Now we try to give some qualitative analyses on the above results. HotSpot missed the root causes of some anomaly cases (see the precision-recall points of HotSpot in Fig. 10) because: a) HotSpot can not calculate the forecast values absolutely accurate due to the inherent limitations of the forecast algorithms and the noises in the dataset; b) we set the *potential threshold* to control the exit of HotSpot to balance accuracy and computational cost, which may cause the method to miss some real root causes, especially when the anomaly cases are in higher layers or they contain more elements. In addition, the reasons why iDice's performance decreases with the increase of the number of anomaly elements is that: a) the first step of iDice, i.e., Impact-based pruning may mistakenly prune some elements; b) the isolation power in iDice may not work well for anomaly cases with more elements. While Adtributor is designed for exploring root causes for anomaly cases in layer one, it did not find root causes for anomaly cases in higher layers in our scenario.

Summary of effectiveness comparison. In summary, Adtributor can only handle anomaly cases in layer one (with a run time of 10 seconds) and iDice can only handle the anomaly cases which have 1 or 2 elements (with a run time of 20 seconds), while HotSpot can handle higher layers and larger number of root cause elements (with a run time of 50 seconds). In reality, the number of elements and the layer of anomaly cases are almost always unpredictable, thus HotSpot is a much better choice than Adtributor and iDice.

G. The Efficiency of HotSpot

In this section, we evaluate the benefits of MCTS and hierarchical pruning in HotSpot in terms of reducing the computational complexity when facing the huge search space. We compare HotSpot versus "HotSpot minus MCTS" and "HotSpot minus hierarchical pruning" methods. The "HotSpot minus MCTS" method means that we employ the hierarchical pruning strategy to search the cuboids layer by layer (from layer one to layer four) and conduct hierarchical pruning. For each cuboid, the method employs the full search method other than MCTS. The "HotSpot minus hierarchical pruning" method employs MCTS for each cuboid to obtain the *BSet*s, and selects the one with the largest *ps* as *RSet*. We do not consider the "full search method" (use neither MCTS nor hierarchical pruning) for it takes too long to run.

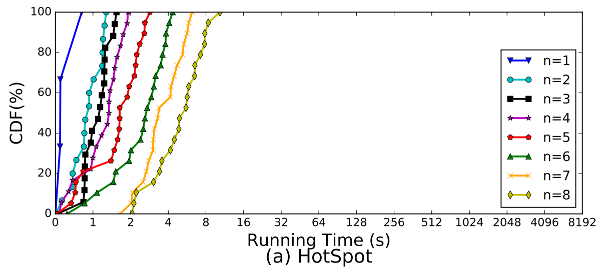
TABLE X: The number of elements for each cuboid when $n = 2$, where n is the number of distinct values in each dimension.

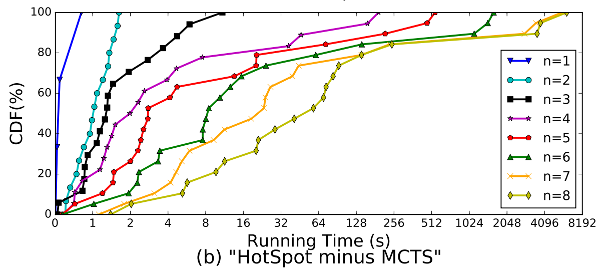
Layer ID	Cuboid (the number of elements in it)					
Layer 1	B_P (2)	$B_I(2)$	$B_D(2)$	$B_C(2)$		
Layer 2	$B_{P,I}(4)$	$B_{P,D}(4)$	$B_{P,C}(4)$	$B_{P,D}(4)$	$B_{P,C}(4)$	$B_{D,C}(4)$
Layer 3	$B_{P,I,D}(8)$	$B_{P,I,C}(8)$	$B_{P,D,C}(8)$	$B_{P,D,C}(8)$		
Layer 4	$B_{P,I,D,C}$ (16)					
Total	80					

Because of the low computational efficiency of "HotSpot minus MCTS" and "HotSpot minus hierarchical pruning" methods, it is prohibitive to evaluate either of the above methods based on the large-scale dataset described in §V-A (hereafter, we collectively refer to this dataset as original dataset). Therefore, we sample the original dataset to obtain new datasets with smaller scale (hereafter, we collectively refer to this dataset as new dataset). Specifically, the new datasets have four dimensions, and for each dimension, there are n distinct values (e.g., Beijing, Shanghai, ..., of B_P) sampled from the original dataset. n can be $1, 2, ..., 8$. E.g., if $n = 2$, the number of elements for each cuboid is shown in Table X.

We injected 20 types of anomaly cases to the new dataset following §V-C. Note that not all the 20 types of anomaly cases exist for every new dataset. For example, when $n = 2$, the anomaly case type "layer one and three elements in each case" does not exist since there are only two elements in each cuboid of layer one. For a new dataset, we injected 400 anomaly cases for each type of anomaly cases, and applied the above three methods to localize anomalies, respectively. We calculated the average running time for each method, and each type of anomaly case. Each method achieved an averaged F-score over 90% in this experiment. Similarly, all the three methods are running on the same server as mentioned in §V-D.

Fig.11 compares the CDFs of the running time of the three methods under different values of n. A point in Fig.11 is the average running time for a specific anomaly case type and a specific value of n. Please note that the x-axis scale is log(2) scale. For each value of n, the running time in Fig. 11 (a) is much smaller than that in Fig. 11 (b) and Fig. 11 (c), which demonstrates that HotSpot is much more computationally efficient than other two methods.





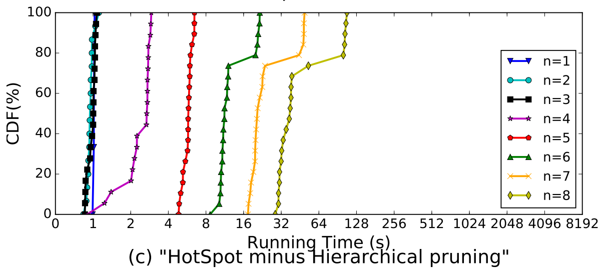


Fig. 11: Comparison of running time of HotSpot, “HotSpot minus MCTS” and “HotSpot minus hierarchical pruning”

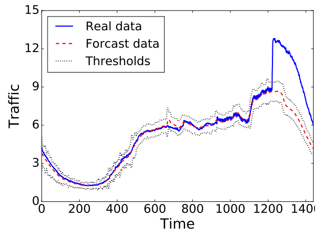
Additionally, the running time for different values of n in Fig. 11 (a) is more centralized than that for different values of n in Fig. 11 (b) and Fig. 11 (c), demonstrating HotSpot’s good robustness in computational efficiency.

VI. Operational Experience

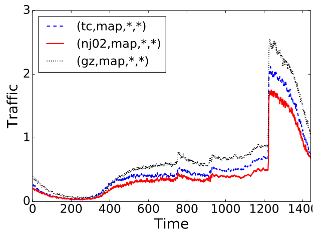
We have implemented and deployed HotSpot in a top global search engine company. We applied HotSpot on various additive KPIs, such as PV, traffic volume, number of online users, and ads revenue. HotSpot has demonstrated its ability to quickly localize the root cause for additive KPIs.

Due to space limitation, we present two operational cases. The data set has four dimensions: DC (11 values), Product (182 values), ISP (7 values) and Server Cluster Name (480 values), and the additive KPIs are Page View and Error Count, respectively. HotSpot spent 10 to 20 seconds in anomaly localization for both cases. For comparison purpose, we asked the operators to manually localize the root causes, and it took operators 1 to 2 hours to do so. That is, HotSpot is about 300 times faster than manual localization, which is typical in our HotSpot *vs.* manual comparisons. Please note that we anonymously the magnitude of the data for privacy, so the value of the data in the following figures are not real.

Case 1: The KPI in this case is the volume of traffic flow of the search engine from the clients. Fig. 12(a)



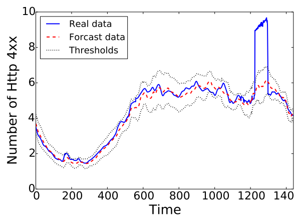
(a) Total



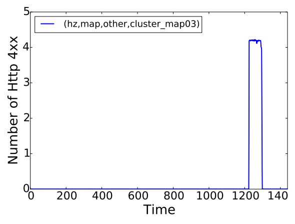
(b) Root cause elements

Fig. 12: Case 1: Page View

shows the actual total traffic values and the forecast ones within some day (the real magnitude is normalized for confidentiality, and case 2 is the same). The measurement interval is one minute, and, thus, there are 1440 intervals for one day. An anomalous sudden increase occurred at the 1223th interval. Fig. 12(b) shows the root cause set which was localized by HotSpot, i.e., $\{(tc, map, *, *); (nj02, map, *, *); (gz, map, *, *)\}$. This result is correct that have been confirmed by operators. The truth of this case is that a faulty configuration of map is updated on the three DCs $(tc, nj02, gz)$. This case further confirms that a root cause set can include multiple elements in the same cuboid. As shown in §V, [1] cannot deal with such cases, and [2] is not as accurate as HotSpot when tackling such cases.



(a) Total



(b) Root cause elements

Fig. 13: Case 2: HTTP Error 4xx Count

Case 2: The measure in case 2 is the number of “HTTP 4xx” errors, e.g., “HTTP 403” and “HTTP 404”, and more details about HTTP status codes can be found in [10]. Fig. 13(a) shows an anomaly of the total number of “HTTP 4xx” occurs at the 1220th point on a day different from case 1.

HotSpot localizes the root cause set to be $\{(hz, map, other, cluster\_map03)\}$, shown in Fig. 13(b). We can see that the element $(hz, map, other, cluster\_map03)$ usually has few “HTTP 4xx” errors, but the error count suddenly increased at the 1220th point that caused the total error count to increase as well. Operators have confirmed that a new application version with a wrong configuration was deployed at the 1220th point, which led to this anomaly.

VII. Discussion

Anomaly localization of multi-dimensional indicator systems is complicated in practice. In this section, we discuss some issues regarding anomaly localization and clarify the scope of HotSpot.

Anomaly detection. As in previous work [1], in HotSpot we assume that anomaly localization is triggered by anomaly detection, and the forecast values output by the anomaly detection is used as input. The selection of anomaly detection algorithms is a problem by itself and is beyond our scope.

Ripple effect has limitations. We propose the potential score based on Ripple effect, but there exist some rare cases that cannot be handled by ripple effect, e.g., if $f(x) = 0$ in Eq. (5). If so, we can extract these elements and analyze them manually, and the remaining ones can still use HotSpot.

HotSpot does not guarantee optimal results. Since both MCTS and hierarchical pruning are heuristic algorithms, HotSpot does not guarantee that it can always find the global optimal result. Even so, it can greatly narrow down the root cause scope and give operators timely advices. Note that, instead of providing just one candidate root cause, HotSpot can provide a ranked list of n candidate root causes to increase the chance that the true root cause is included in the list.

VIII. Related Work

There are many previous works in root cause localization in various contexts. Pinpoint [11] diagnoses the root causes of large, dynamic Internet services (e.g., TCP and HTTP failures) employing clustering analysis. SCORE [12], Shrink [13], and [14] focus on localizing IP network failures in an IP-over-optical tier-1 backbone using a Shared Risk Link Group (SRLG) model. They try to identify a smallest set of risk groups that can explain the failures, which actually used the succinctness concept. Sherlock [15] focuses on localizing the root causes of performance problems among numerous dependencies of network elements in large enterprise networks, using packet traces, traceroute measurements, and network configuration files. Argus [16] detects services anomalies from an ISP's perspective, aiming to localize the users with bad performance. It uses a hierarchical data structure to aggregate users with common attributes and localize each user groups' performance. ABSENCE [17] detects service disruptions in mobile networks using aggregated customer usage data. Its hierarchization is also a tree structure, which is different from our work. FOCUS [18] is an approach on determining the long-term bottlenecks in multi-dimensional logs.

Our work is different from all the above studies in terms of both the problem definition and the solution being used. On the one hand, our problem is focused on fine-grained anomaly localization on multi-dimensional systems. The data values of the system are additive and the demand of the result is succinctness. None of the above studies is similar with this. On the other hand, most of previous works apply intuitive experienced empirical methods to simplify the complex problems, while in our paper, we propose an innovative fundamental idea (with a very high complexity) at first, then we employ MCTS and hierarchical pruning strategies to realize the idea in a very reasonable time, and this effectively balances the efficiency and effectiveness.

There are three previous works that are closely related to our work. iDice [2] and Adtributor [1] tackle a similar problem. They are compared with our approach and discussed in detail in Section V-F. [19] tackling the anomaly detection and localization in a ISP setting, and its concept of E2E instance is similar to the element. However, the paper is mainly focused on anomaly detection. For anomaly localization, they only sketched an idea (applying association rule mining algorithm) in three short paragraphs without sufficient algorithm details. Nonetheless, this method requires different set of parameters (minimum support and minimum confidence) for different types of cases defined in V-C. However, it is impossible to know in advance which type the case belongs to. Due to this shortcoming and lack of algorithm details, we conclude that it is infeasible to do a fair comparison with [19] in the evaluation section.

IX. Conclusion

For an additive KPI with multi-dimensional attributes, it is a hard problem to localize the overall KPI's anomaly to the root cause, which is one (or more) combination of attribute values in multiple dimensions. Firstly, we consider this anomaly localization as a search problem with a huge space. To deal with the huge search space, our proposed framework, HotSpot, adopts the MCTS approach (the first time in anomaly localization literature) whose action value is our novel potential score based on the "ripple effect", which captures how anomalies propagate from the root cause throughout the aggregation hierarchy. In addition, we propose a hierarchical pruning approach to further reduce the search space. Our experiments based on the data from a real-world search engine show that HotSpot achieves much better accuracy than previous approaches. Our operational experiences show that HotSpot can reduce the localization time from about more than 1 hour in manual efforts to less than 20 seconds, and that HotSpot is an approach generally applicable to the anomaly localization for additive KPI metrics.

X. ACKNOWLEDGMENT

This work is supported by the State Key Program of National Science of China under grant No. 61233007, the National Natural Science Foundation of China (NSFC) under grant NO. 61472214, 61472210, the National High Technology Development Program of China (863 program) under grant No. 2013AA013302, the National Key Basic Research Program of China (973 program) under grant No. 2013CB329105, the Tsinghua National Laboratory for Information Science and Technology key projects, the Global Talent Recruitment (Youth) Program, and the Cross-disciplinary Collaborative Teams Program for Science & Technology & Innovation of Chinese Academy of Sciences-Network and system technologies for security monitoring and information interaction in smart grid.

References

[1] 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.

[2] Q. Lin, J. Lou, H. Zhang, and D. Zhang, “idice: problem identification for emerging issues,” Proceedings of the 38th International Conference on Software Engineering. ACM,, pp. 214–224, 2016.

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

[4] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, “A survey of monte carlo tree search methods,” IEEE Transactions on Computational Intelligence and AI in Games, vol. 4, no. 1, pp. 1–43, 2012.

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al., “Mastering the game of go with deep neural networks and tree search,” Nature, vol. 529, no. 7587, pp. 484–489, 2016.

[6] 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.

[7] L. Kocsis and C. Szepesvári, “Bandit based monte-carlo planning,” in European conference on machine learning. Springer, 2006, pp. 282–293.

[8] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” Machine learning, vol. 47, no. 2-3, pp. 235–256, 2002.

[9] A. Soule, K. Salamatian, and N. Taft, “Combining filtering and statistical methods for anomaly detection,” in Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement. USENIX Association, 2005, pp. 31–31.

[10] https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#4xx_Client_errors.

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

[12] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, “Ip fault localization via risk modeling,” in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2. USENIX Association, 2005, pp. 57–70.

[13] 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.

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

[15] 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,” in ACM SIGCOMM Computer Communication Review, vol. 37, no. 4. ACM, 2007, pp. 13–24.

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

[17] 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.

[18] 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 INFOCOM, 2016 Proceedings IEEE. IEEE, 2016.

[19] 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," in INFOCOM, 2016 Proceedings IEEE. IEEE, 2016.