# Locating Anomaly Clues for Atypical Anomalous Services: An Industrial Exploration

Guoping Rong*, Hao Wang*, Shenghui Gu*, Yangchen Xu*, Jialin Sun†, Dong Shao*, He Zhang*
*State Key Laboratory for Novel Software Technology,
Software Institute, Nanjing University, Nanjing, China
ronggp@nju.edu.cn, wanghao.stu.nju@gmail.com, shenghui.gu@smail.nju.edu.cn,
mf1932211@smail.nju.edu.cn, dongshao@nju.edu.cn, hezhang@nju.edu.cn
†Meituan, Shanghai, China
jialinsun815@gmail.com

**Abstract**—Continuity and steadiness are vital for services with massive users, which requires the anomalies of services should be detected and resolved in a timely manner. Our previous work proposed a tool, namely *ImpAPTr (Impact Analysis based on Pruning Tree)*, to identify the combination of multiple dimensional attributes as the clues leading to the root cause of service anomalies. However, *ImpAPTr* applies a threshold driven strategy, i.e. it needs to be triggered by a $\geq 0.05\%$ drop of the success rate of the service calls (abbr. *SRSC*), which may face problems in an atypical yet pervasive situation in field application. For example, the combination of trivial anomalies (i.e. each causes a drop less than $0.05\%$ to *SRSC*) can lead to a far more than $0.05\%$ drop on *SRSC*. Besides, a suitable threshold is usually hard to be determined, etc. To address these problems, we propose a new method, namely *ImpAPTr+* in this paper to free the constraint of the $0.05\%$ threshold. The basic idea is to involve time dimension and identify clues across multiple time intervals of data. We performed evaluation on three typical methods (i.e. *ImpAPTr+*, *R-Adtributor* and *Squeeze*) with both production environment dataset and simulation dataset. The former dataset is directly retrieved from the service monitoring data in *Meituan*, one of the largest on-line service providers worldwide. The latter dataset is fabricated also using the monitoring data from the same company. The results indicate: (1) *ImpAPTr+* outperforms previous approaches to a large degree in terms of accuracy. (2) Both *ImpAPTr+* and *R-Adtributor* are able to find proper clues within seconds. (3) *ImpAPTr+* tends to find proper clues with shorter time intervals (i.e. less data), which implies that the method is more suitable for near real-time monitoring scenarios.

**Index Terms**—On-line service monitoring, Anomaly clues locating, Multiple dimensional attributes

## 1 Introduction

Nowadays, tremendous Internet companies provide diversified services through various on-line software systems. The continuity and reliability of services are thus critical since even a slight decline of the success rate of service calls (abbr. *SRSC*) may have impacted a large number of users already. In this sense, it is important to monitor *SRSC*, detect and resolve anomalies impacting *SRSC* in a timely manner. Several Application Performance Management (APM) systems have been adopted to perform such monitoring, for example, CAT[1], Prometheus[2], Pinpoint[3], SkyWalking[4] and Zipkin[5] are all popular APM tools with massive users. Nevertheless, monitoring *SRSC* only cannot eliminate an anomaly. We also need to identify and address its root cause, which is not an easy job in many on-line systems. The main reason is that current software services are more and more distributedly deployed and ubiquitously accessed than ever (e.g., through various devices, regions, etc.), therefore an anomaly such as a Declining Success Rate ($DSR_\Delta$) may occur due to complex reasons in a production environment. Although most APM tools could capture the trace of service calls and the corresponding responses, the captured information may contain multi-dimensional attributes (e.g., City, ISP, Software Version, etc.) with multiple values for each attribute, for example, the ISP could be T-Mobile, Vodafone, CMCC, etc. Take a combination $\mathscr{S}$(4G, *ABC*, CMCC, 1.0.1, iOS) for example, it denotes that a service call is from an iOS device using *CMCC 4G* network in City *ABC* and the APP version is 1.0.1. Apparently, $\mathscr{S}$ may provide useful clues to identify the root cause for an anomaly. However, locating $\mathscr{S}$ is not easy in a production environment, given that there might be tens of thousands of possible combinations and the time slot allowed for locating $\mathscr{S}$ is usually slim. Note that $\mathscr{S}$ may only contain partial dimensional attributes of the trace information for a service call and response.

Fig. 1 depicts a typical example of what happens when $DSR_\Delta$ occurs and *SRSC* recovers later on. The data behind this figure is extracted from the actual production environment in *Meituan*, one of the companies with the largest scale on-line business worldwide. In the first time interval $t_1$ (i.e. 07:55∼08:00), for example, the *SRSC* is $99.52\%$ with $30085$ requests to this service. However, in the second interval $t_2$ (i.e. 08:00∼08:05), the *SRSC* suddenly drops to $98.65\%$, indicating a $DSR_\Delta$ of $0.87\%$ and nearly 402 failed service responses. In the comparison of $t_1$ and $t_2$, the number of requests in $t_2$ is $29845$, which is quite similar to that in $t_1$.

* *He Zhang is the corresponding author.*

1. https://github.com/dianping/cat
2. https://prometheus.io/
3. https://pinpoint-apm.github.io/pinpoint/
4. https://skywalking.apache.org/
5. https://zipkin.io/

We can observe that the *SRSC* recovers to $98.99\%$ in the third interval $t_3$ (i.e. 08:05∼08:10), while the number of requests in this interval declines to 28390. In the fourth interval $t_4$ (i.e. 08:10∼08:15), the *SRSC* is $98.81\%$, indicating a $DSR_\Delta$ of $0.18\%$. In the fifth time interval $t_5$ (i.e. 08:10∼08:15), the *SRSC* slightly declines to $98.78\%$ ($DSR_\Delta$ $0.03\%$) while the request number is 33655. Note that there are nearly 410 failed service responses during $t_5$, which is even more than the failures in $t_2$.

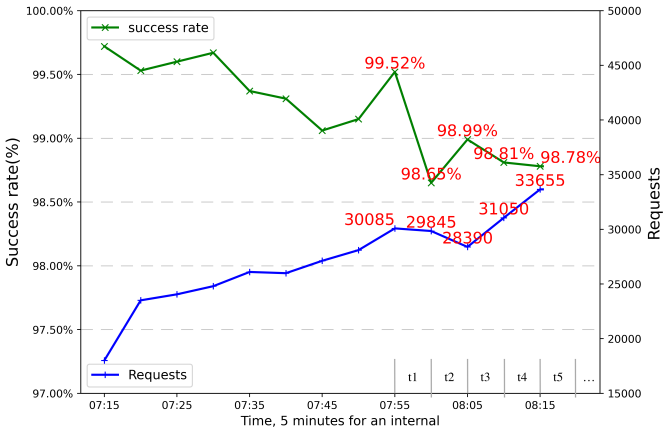



Fig. 1: Trend of success rate of service calls

In our previous study, we proposed a method (*ImpAPTr*) to identify the valid combination of multiple attributes which serves as a clue leading to the root causes for a certain anomaly [1]. However, *ImpAPTr* requires a predefined threshold of $0.05\%$ $DSR_\Delta$ in current time interval (normally 5 minutes) compared to its immediately previous time interval to confirm an anomaly and then be triggered to perform analysis and discriminate the most likely clues as well. Although *ImpAPTr* has been successfully adopted in *Meituan*, it has encountered several new challenges during its execution.

- Firstly, a suitable threshold is usually determined by many factors which is thus difficult to be decided. In [1], *ImpAPTr* applied a threshold of $0.05\%$ which is required by the operations management based on their experience and department policy. However, a fixed threshold may lack of both efficacy and efficiency under different contexts such as service types, user groups, timing, etc. For example, addressing the anomalies (even with larger than $0.05\%$ $DSR_\Delta$) to a non-core service during off-peak periods may not bring noticeable benefits for business.
- Secondly, as shown in Fig. 1, although a declining trend to *SRSC* is observable, which has already implied the existence of anomalies, yet the $DSR_\Delta$s in multiple time intervals might not exceed the predefined threshold (i.e. $0.05\%$). While this phenomenon is not rare in the production environment, the challenge is however to determine a proper timing to run *ImpAPTr*.
- Last but not least, *ImpAPTr* cannot handle a combination of trivial anomalies, i.e. each anomaly results in less than $0.05\%$ drop on *SRSC*, but as an accumulation may cause a $DSR_\Delta$ much larger than $0.05\%$. As a matter of fact, under this situation, even *ImpAPTr* will be triggered to perform analysis, it will not generate any proper clues since *ImpAPTr* also uses $0.05\%$ $DSR_\Delta$ as the criteria to prune the element tree (a tree structure derived from the combinations of multi-dimensional attribute values, cf. Section 3.3.3). On the other hand, for the services with massive users, a $DSR_\Delta$ of $0.05\%$ (even less) may have impacted a large number of users already and the number may continue to grow if the root cause could not be addressed quickly.

It seems that a reactive strategy (i.e. a pure threshold driven policy to run *ImpAPTr*) may not be able to bring as much as possible benefit to the business in practice, thus a proactive strategy turns to be desired to strengthen its capability regarding observability [2] to the complex systems supporting various on-line services in this company. Motivated by this urge, we extended our previous study [1] and proposed a redesigned method, namely *ImpAPTr+* to free the constraints derived from the $0.05\%$ threshold. To be specific, we remove any $DSR_\Delta$ threshold and involve time factor into *ImpAPTr+* to identify a clue using the monitoring data (i.e. the records of service calls captured by the APM tool) across multiple time intervals. Compared with the previous study [1], the major extension and differences are listed in TABLE 1. As shown in this table, the text in bold describes the differences between this work and our previous study [1] from four aspects, i.e. targeting problem, locating algorithm, research process and implications, meaning that a new study to address new problems with new solutions will be presented in this paper. Meanwhile, the two studies also share certain similar elements, e.g., the characteristics of the clues (i.e. multiple dimensions), the requirement of timeliness, the evaluation strategy (i.e. both production dataset and fabricated dataset involved), and one criterion to perform evaluation (i.e. the time required to analyze the data from one time interval), as depicted in TABLE 1 with regular-font text.

In general, the contribution of our study can be highlighted as follows.

- Firstly, a new method (*ImpAPTr+*) to identify clues leading to the root causes of anomalies has been designed which is able to work without any predefined threshold on $DSR_\Delta$. In this way, *ImpAPTr+* can perform proactive and persistent analysis to detect clues associated with even trivial anomalies. Based on their contribution to $DSR_\Delta$, potential clues are sorted to imply the most worthy investigation direction.
- Secondly, we conduct empirical evaluations on three methods, i.e. *ImpAPTr+*, *R-Adtributor* and *Squeeze* with both production environment dataset and new designed simulation dataset to investigate their performance extensively. The results indicate that *ImpAPTr+* outperforms the other two methods in terms of accuracy. Both *ImpAPTr+* and *R-Adtributor* can locate a proper clue within seconds. Meanwhile, *ImpAPTr+* tends to use less data (time intervals) to locate a proper clue, which to a certain degree implies

TABLE 1: A brief comparison between this study and the previous study [1]

| Aspects | Previous study (*ImpAPTr*) | Current study (*ImpAPTr+*) | Elaborated in |
|---|---|---|---|
| Targeting problem | · **To identify the most likely clues leading to the root causes of anomalies to services using a predefined and fixed threshold (i.e. 0.05% $DSR_\triangle$)**<br>· The clues are combinations of multi-dimensional attributes.<br>· The value of each multi-dimensional attribute should be determined in a timely manner from massive service calls. | · **To identify the most likely clues leading to the root causes of anomalies to services without any predefined threshold**<br>· The clues are combinations of multi-dimensional attributes also.<br>· The value of each multi-dimensional attribute should also be determined in a timely manner from massive service calls. | Section 1, 2, 3.2 and 7. |
| Locating algorithm | · **A breadth-first traversal on the element tree which uses *CP (Contribution Power)*, *IF (Impact Factor)* and *DF (Diversity Factor)* as the differentiate factors to rank potential clues.**<br>· **0.05% $DSR_\triangle$ is not only a threshold to determine an anomaly, but also a criterion to prune the corresponding element tree.** | · **A breadth-first traversal on the element tree which uses *Ranking Score* as the differentiate factor to rank potential clues.**<br>· ***Ranking Score* is calculated using the Euclidean Distance on *IF* and *DF*, which further transforms into *Weighted Ranking Score* to reflect the historical effect for a potential clue.** | Section 4. |
| Research process | · Both real production dataset and fabricated dataset are used in evaluation.<br>· **We applied $\geq 0.05\%$ $DSR_\triangle$ as the criterion to prepare both datasets.**<br>· **Both *accuracy* and *efficiency* are evaluated.** | · Both real production dataset and fabricated dataset are used in evaluation.<br>· **A time box including 6 intervals is determined based on the observation to the data characteristics from the production dataset, which further applied in dataset fabrication.**<br>· **No criterion on $DSR_\triangle$ is required to select the real production dataset or prepare the fabricated dataset. However, to perform the evaluation, we planted relatively trivial $DSR_\triangle$ (i.e., 0.01% to 0.03%) to the fabricated dataset.**<br>· **Both *accuracy* and *efficiency* are evaluated. However, due to different targeting problem, *efficiency* is evaluated from three perspectives, i.e.,** time to analyze the data from one time interval, **time to identify a clue correctly, and timing to identify a clue correctly.** | Section 5 |
| Implications | · **Using $0.05\%DSR_\triangle$, a reactive strategy is able to identify useful clues with multi-dimensional attributes leading to service anomalies in a matter of seconds in real production environment in *Meituan*.** | · **Without any threshold constraint, *ImpAPTr+* supports a proactive identification to useful clues with multi-dimensional attributes leading to service anomalies in dozens of seconds in real production environment in *Meituan*.** | Section 1, 6 and 7 |

that *ImpAPTr+* can be deployed in the production environment to provide a near real-time monitoring to the *SRSC* for a service.

## 2 Related Work

Driven by business needs, detecting service anomalies, locating and addressing their root causes attract many researchers' interests. Research on anomaly detection typically concentrates on detecting or confirming the occurrence of anomalies to on-line services. Both statistics based methods (e.g., [3], [4], [5], [6]) and machine learning based methods (e.g., [7], [8], [9], [10], [11], [12], [13], [14]) have been extensively explored and investigated in recent years. Nevertheless, as discussed above, the targeting problem we need to address is to timely locating a combination of multi-dimensional attributes as the clues/root causes associated with an anomaly regarding *SRSC* to on-line services. As reflected in Fig. 1, for on-line services with massive users, it is nearly impossible to achieve 100% success rate of service calls and responses, implying a constant occurrence of anomalies, though it may not be economical to dig the root causes out sometimes.

Locating and addressing root cause (we use this term in this section) of anomalies also attracted considerable research effort. Based on the number of dimensional attributes involved, there are two types of research, i.e. single attribute and multiple dimensional attributes. Meanwhile, perhaps driven by the proliferation of on-line services with massive users, trivial anomalies began to attract researchers' attention recently.

### 2.1 Single attribute

This type of studies focus on locating the root cause using single attribute. Some experimental cases are provided in [15] on locating causes of anomaly pertinent to CDN (Content Delivery Network) and routers via the DAG (Directed Acyclic Graph), which is formed by tracing logs using the "X-trace" framework [16]. Lou et al. [17] and Chow et al. [18] focus on the root cause of the error logs of all components within the tracing path by establishing a causal graph. In [19], researchers diagnose the network problems occurring on the network components based on a decision tree [20]. Shrink [21], [22] and SCORE [23] are two typical methods focusing on the fault locating in IP networks based on risk models and SRLGs (Shared Risk Link Groups) [24]. FOCUS [25] is designed to find out the root cause of HSRT (High Search Response Time), where images, browser engines, ISP and other factors have been explored. The root cause locating of performance related problems has been studied in [26], [27], [28], [29], [30], [31] for different business contexts, and there are also plenty of studies [32], [33], [34], [35], [36], [37] on root cause locating for other issues such as network problems and system-log related anomalies.

### 2.2 Multiple dimensional attributes

Methods dealing with multiple dimensional attributes can be further divided into two types based on the measures applied, i.e. basic measure (e.g., the number of service calls) and derived measure (e.g., *SRSC* in this paper).

*Adtributor* [38] works with both types of measures but can only perform locating using one dimensional attribute at a time. As the improved version of *Adtributor*, *R-Adtributor* [39] is proposed to execute recursive calls based on the results of *Adtributor* to address this limitation. However, since the recursive depth is hard to be determined beforehand, the results may be incorrect. *HotSpot* [40] can only work with basic measures to locate root cause of multi-dimensional attributes. It uses Monte Carlo Search Tree [41] to reduce the search space. *iDice* [42] also works for basic measures and multi-dimensional attributes. *Apriori* [43] can deal with derived measures and multi-dimensional attributes, but the running time is always too long to be adopted in the production situation described in this paper. *Squeeze* [44] improves *HotSpot* to deal with derived measures and avoids omitting some important elements which may be pruned in the original *HotSpot* approach. Ahmad et al. also proposed an machine learning-based approach that detects and locates root cause of end-to-end performance degradation along with four dimensional attributes in cellular services [45].

### 2.3 Trivial anomaly detection and analysis

Although there is no widely accepted criteria to define a "trivial anomaly", the ever-growing scale of on-line services has formed the necessity to concern trivial, or subtle anomalies, as even trivial anomalies can affect a considerable number of users. Another important reason is that trivial anomalies can be very common in real-world environment, and the potential long tail effect, regarding the existence of similar phenomenon in cloud services' response times [46], may further magnify the trivial anomalies' impact on user experience. Notably, the presence of trivial anomalies has been mentioned in [47], [48], [49]. Nevertheless, how to locate the root cause of the detected trivial anomalies has not been studied in these work.

As a rule of thumb, the detection or analysis of trivial anomalies needs more information to generate meaningful differences or patterns. A technology called contrast data mining [50] has thus been applied to perform root causes analysis [51], [52]. Contrast data mining could discover contrast patterns which describe significant differences between datasets. Intuitively, persistently identifiable and significant differences should exist before contrast patterns can be recognized. To achieve this, researchers in [52] differentiates the test group (traces with bug) from the control group (traces without bug) to support pattern recognition. The framework proposed in [51] extracts basic measures from the structured logs to support frequent item-set mining. When applying to non-discrete anomaly indicators such as latency, acceptable values of latency should also be manually differentiated from the non-acceptable values. However, the complicated production environment and ever-changing *SRSC* (a non-discrete derived measure) make it difficult to have a relatively healthy dataset to be used as the contrast basis. For example, a combination $\mathscr{S}$ might contain both successful and failed requests in one time interval. Besides, the same $\mathscr{S}$ may appear in one time interval but disappear in the following time intervals. To provide more information, continuous monitoring which captures the dynamics of the underlying system and is able to involve more information from the time dimension is thus believed to have a potential to be widely used for root cause analysis [53].

With reference to [44] and our previous work [1], we list several potential methods in TABLE 2, to the best of our knowledge. However, the characteristics of the targeting problem described in this paper (i.e. multiple dimensional attributes involved, continuous impacts on a derived measure such as $DSR_{\triangle}$, and relatively short time to perform the analysis, etc.) cast challenges to develop applicable solutions. According to TABLE 2, only *Squeeze* [44], *R-Adtributor* [39] and *ImpAPTr* [1] have the potential to be valid solutions. Nevertheless, as mentioned above, *ImpAPTr* needs a predefined threshold, which is not ready for out-of-the-box application.

TABLE 2: A brief comparison to existing root cause location methods

| Method | SA or MA[1] | BM or DM[2] | Time Cost |
|---|---|---|---|
| *HotSpot* [40] | MA | BM | sometimes long |
| *Adtributor* [38] | SA | BM&DM | very short |
| *R-Adtributor* [39] | MA | BM&DM | short |
| *iDice* [42] | MA | BM | very short |
| *Apriori* [43] | MA | BM&DM | always too long |
| *Squeeze* [44] | MA | BM&DM | short |
| *ImpAPTr* [1] | MA | DM | short |

[1] SA: Single Attribute; MA: Multiple Attributes
[2] BM: Basic Measure; DM: Derived Measure

## 3 Problem Statement

In general, compared to *ImpAPTr* [1], the *ImpAPTr+* proposed in this study performs different analysis with the same information collected by the CAT system. Therefore, we reuse most content regarding the description of the targeting problem in [1] here. Meanwhile, differences on threshold requirement, processing tool, etc. are also elaborated in this section.

### 3.1 "Root Cause" or "clue"

Many studies use "root cause" to indicate the reason for a certain anomaly which is the objective for many locating algorithms. However, for many cases in practice, many times the genuine reason may be covered and concealed beneath a thick layer of symptoms. For example, the combination of (SH, 4G) could either be some malfunction of 4G network occurred in the region of SH (which we may not know why yet) or some issues existing in the source code which make the corresponding service unstable during the turbulence of network data. While the combination of (SH, 4G) in the former scenario may justify the literally meaning of "root cause", it only represents the clues leading to a root cause in the latter scenario. In this sense, we use "clues" instead in this paper, meaning a worthy direction for engineers to carry out further investigation. As a matter of fact, along with the popularity of cloud computing, it is commonly more difficult (if not impossible) to locate the genuine reasons for anomalies since the infrastructure can be complicated and transparent to the operations staff of business systems [54], [55], [56]. In this sense, finding valid clues should be taken as a more practical objective for the

TABLE 3: Terms and concepts used in this study

| Terms & Concepts | Definition | Notation | Example |
|---|---|---|---|
| Dimensional Attribute | The category of information (data) contained in a service call | – | Network, Connect-Type, Platform, ISP, City, etc. |
| Attribute Value | The value of a certain dimensional attribute | – | Take Network for example, WIFI, 3G, 4G, 5G, etc. |
| Element | A vector of attribute values | e = ($*$[1], $*$, $*$, $*$, $*$) | ($*$, Type1, $*$, $*$, $*$), (3G, Type2, $*$, Telecom, $*$), etc. |

[1] $*$ denotes any valid value regarding a certain dimensional attribute.

identification or locating algorithms in cloud computing, which becomes a clear trend nowadays. Apparently, a valid clue should contain valuable information about the right direction to explore the root cause of anomaly without misleading information.

### 3.2 Background

*Meituan* is one of the largest on-line services providers worldwide. Through *DP* system, *Meituan* provides various on-line services to tens of millions of users simultaneously. To guarantee the healthiness of the *DP* system, engineers developed and adopted CAT, an APM system to monitor the status of all the services. Or rather, to monitor more than ten thousands of services after a major technological transformation to microservices architecture. One of the key tasks regarding service monitoring is to dig out the clues leading to the corresponding root causes of certain anomalies pertinent to various business Key Performance Indicators (KPIs) from massive volume of tracing data. Our research in the first stage results in a method (*ImpAPTr* [1]) to identify a single clue pertinent to anomalies with more than $0.05\%$ $DSR_\Delta$ within seconds. Nevertheless, *ImpAPTr* encountered several problems in field application. For example, it cannot cope with a $DSR_\Delta$ constituted by multiple trivial anomalies even the $DSR_\Delta$ is much larger than $0.05\%$ but each anomaly causes a $DSR_\Delta$ less than $0.05\%$. Moreover, a predefined threshold (i.e. $0.05\%$ $DSR_\Delta$) is hard to be determined and adapted to various on-line services. Therefore, we extensively carry out this study to identify the clues associated with those persistent anomalies without the predefined threshold.

TABLE 4: Dimensional attributes and their valid values

| Dimensional Attributes | Notation | Legal Values |
|---|---|---|
| Network[1] | $N$ | Unknown, WIFI, 2G, 3G, 4G |
| Connect-Type[*2] | $C$ | Type0, Type1,. . . , Type7, Type8 |
| Platform[3] | $P$ | Unknown, Android, IOS |
| ISP[4] | $I$ | CMCC, Telecom, T-Mobile, Other, etc. |
| City[5] | $Y$ | Shanghai, Beijing, etc. |
| App-source | $S$ | App1, App2, App3, App4, etc. |
| App-version | $V$ | 1.0.0, 10.0.2, 10.1.2 etc. |

[*] e.g., https, http persistent connection and some self-defined types. For ease of description, we use Type0, Type1, etc. here.
[1,2,3,4,5] These are the 5 dimensional attributes used most by the operations staff in Meituan.

### 3.3 Problem description

TABLE 3 first defines some key concepts and terminologies used in this study, in which Dimensional Attributes can be taken as the categories of information contained in a service call which will be captured and recorded by APM systems. In practice, there are usually seven dimensional attributes being adopted in the APM system in *Meituan* at present. TABLE 4 presents the seven dimensional attributes and their corresponding options which are extracted from the APM system (i.e. CAT system). Among these dimensional attributes, the first five in TABLE 4 are the most investigated dimensional attributes by the operations staff in *Meituan* if a $DSR_\Delta$ occurs, meaning that they will check these 5 attributes and all their pairs. Quite often, these time-consuming investigations achieved nothing since the valid clue may be buried in a huge number of possible pairs. In this study, we also use these 5 attributes for evaluation, i.e. `Network`, `Connect-type`, `Platform`, `ISP`, and `City`. Take `Network` for example, the possible options for this attribute can be 'Unknown', 'WIFI', '2G', '3G' and '4G'. Examples of the actual information regarding these five dimensional attributes are presented in TABLE 5. As shown in TABLE 5, for each request, other than the values for these attributes, the timestamp and status code (to indicate a successful call or a failed call) are also recorded. As discussed above, $DSR_\Delta$ and *SRSC* require key information such as the status of service call and the number of service calls for each status. We define several key metrics and one operation as the following.

TABLE 5: Examples of the request information organized as the top 4 dimensional attributes.

| Timestamp | (Network, Connect-Type, Platform, ISP, City) | Code |
|---|---|---|
| 06:00 | (4G, Type2, Android, Mobile, SH[#]) | 200[*] |
| 06:00 | (4G, Type2, IOS, Unicom, JS[#]) | 100 |
| 06:00 | (WIFI, Type1, IOS, Telecom, SH) | 101 |
| ... | ... | ... |
| 06:05 | (4G, Type2, Android, Mobile, SH) | 200 |
| 06:05 | (WIFI, Type1, IOS, Telecom, JS) | 102 |
| 06:05 | (WIFI, Type2, Android, Telecom, JS) | 200 |
| ... | ... | ... |

[*] 200, the code for a successful service call, otherwise, failed.
[#] Both "SH" and "JS" are the region names.

#### 3.3.1 Fundamental metrics

***Combination of dimensional attributes*** ***($\mathscr{S}$)*** defines the set of dimensional attributes that as a combination (a.k.a. clue) may lead to the root cause of an anomaly of a running service. We set up $\mathscr{S}$ as an element $e$, as follows.

$$e = (n, c, p, i, y),$$
$$(n \in N \text{ or } n = *), (c \in C \text{ or } c = *), (p \in P \text{ or } p = *),$$
$$(i \in I \text{ or } i = *), (y \in Y \text{ or } y = *)$$
$$N = \{N_0, N_1, \ldots, N_4\}, C = \{C_0, \ldots, C_8\}, P = \{P_0, P_1, P_2\}, \quad (1)$$
$$I = \{I_0, I_1, \ldots, I_7\}, Y = \{Y_0, Y_1, \ldots, Y_{309}\} \quad (2)$$

where the wildcard '$*$' can free one or more constraints derived from certain dimensional attributes (as presented in TABLE 4). Apparently, $e$ can be taken as an instance of $\mathscr{S}$. As discussed above, we only involve five dimensional attributes, i.e. $N$, $C$, $P$, $I$ and $Y$. In *Meituan*, 5 options for $N$, 9 options for $C$, 3 options for $P$, 8 options for $I$, and 310 options for $Y$, respectively.

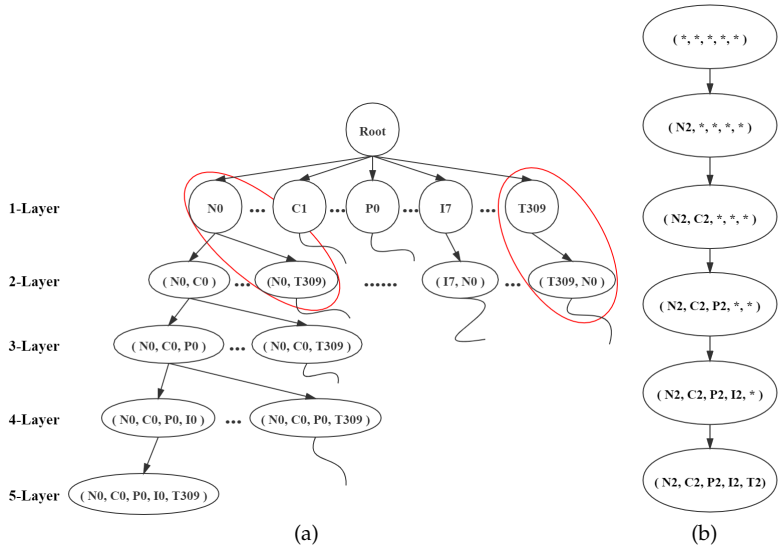



Fig. 2: All elements formed with 5 dimensional attributes. (a) The element tree. (b) A complete extending process from root node.

To support *ImpAPTr+*, we reuse the same metrics defined in [1]. In short, *SRSC* measures the success rate of all service calls within a time interval under the constrain of an element $e$. And $DSR_{\Delta}$ measures the degree to which the *SRSC* of an element $e$ in a certain time interval ($T_i$) is less than that of its previous interval ($T_{i-1}$).

#### 3.3.2 Operation

To support analysis, ***Aggregation (AGG)*** is still used in *ImpAPTr+*, which remains the identical definition in [1]. In general, $AGG$ is defined as the operation to calculate the total value of a certain metric (e.g., number of requests, number of successful requests, etc.) within a time interval ($T_i$), given a predefined $e$. However, to support ***Aggregation***, we use the dataframe tool[6] instead of a high dimensional array in our previous study [1] to save memory. Through the dataframe tool, we can remove the limitation for the number of dimensional attributes involved when running *ImpATPr+* at the expense of more time required to identify clues.

#### 3.3.3 Element tree

To explore the clues leading to a certain anomaly, we have to take all dimensional attributes and their combinations as well for further analysis. As discussed above, an element $e$ is used to represent one of the combinations of dimensional attributes. Therefore, the number of elements is thus huge, theoretically. To portray a concept, we can construct an element tree. As shown in Fig. 2(a), with the five dimensional attributes and their values, we obtain a 5-layer element tree with 335 elements on the first layer, 7,973 on the second, 69,961 on the third, 258,690 on the fourth and 334,800 on the fifth, respectively. As a result, there may potentially be 671,759 elements need to be explored to identify the root cause for a certain anomaly. Apparently, seven dimensional attributes in total will create an even huger number of elements.

6. https://pandas.pydata.org/

## 4 Clue Identification Algorithm

In general, a clue associated with anomalies can be identified with a combination of dimensional attributes (i.e. element $e$) which continuously contribute to produce $DSR_{\Delta}$ (even very small) across multiple time intervals. The challenges are twofold, i.e. the relatively huge search space of elements and the method to quantify the impacts of different elements. In this section, we elaborate the identification algorithm and the rationale behind.

### 4.1 Quantifying impacts

We still keep *Impact Factor* and *Diversity Factor* used in [1] to measure the impacts of an element $e$ towards a $DSR_{\Delta}$. Besides, a new index, namely *Ranking Score* is applied in the new method. In short, *Impact Factor (IF)* measures the degree to which a particular element (e.g., $e_0$) impacts the *SRSC* within a time interval, while *Diversity Factor (DF)* measures the degree of change to the *SRSC* between two adjacent time intervals using the relative entropy calculated by the Jensen-Shannon (JS) divergence [57].

As discussed above, an element owning relatively smaller *IF* and larger *DF* holds larger possibility to be an effective clue. So we designed the *Ranking Score* (*RS*) to measure the combination impact derived from both the two metrics. The first step is to normalize the two metrics using *MIN_MAX* method, and transform them to [0, 1]. We need to select the element with smaller *IF* and larger *DF*, so the normalized *IF*(*nIF*) is "1− *MIN_MAX(IF(e))*" and the normalized *DF*(*nDF*) is "*MIN_MAX(DF(e))*". Then the *Ranking Score* can be calculated using the Euclidean Distance as Equation 3.

$$RS(e) = \sqrt[2]{(nIF)^2 + (nDF)^2} \tag{3}$$

To identify clues pertinent to anomalies is to find the elements shown in continuous time intervals which are related to failed requests. Therefore, we defined *Weighted*

*Ranking Score* (*WRS*) to reflect the historical effect for a certain clue. The rationale behind is that the more often a clue occurred in the near historical time intervals, the higher *WRS*. The equation is thus defined as follows.

$$WRS(e) = RS(e) * (1 + \frac{p}{total}) \quad (4)$$

where $p$ and $total$ respectively represent the occurrences of $e$ and the total number of occurrences within the near past time window (i.e. certain number of time intervals).

### 4.2 Reducing search space

As discussed above, the search space can be rather huge to be dealt with in a timely manner. Therefore, we need to reduce the search space. Several strategies are listed as follows.

Redundant Elements: Since $e$ is a combination of dimensional attributes, the order of the attribute values does not matter. Therefore, as Fig. 2(a) depicts, two nodes (i.e. $(T_{309}, N_0)$ and $(N_0, T_{309})$) in the two red cycles on the second layer can be categorized as redundant nodes (elements). Obviously, only one of the nodes and its subtrees need to be reserved.

Positive Impacts: According to the discussion above, the *Impact Factor* represents the impact of an element on the overall $DSR_\Delta$. While there might be a positive *Impact Factor* for a given element $e$, which means that this element $e$ is related to the decrease of a $DSR_\Delta$. Since we are seeking elements related to the increase of a $DSR_\Delta$, we can also remove the elements with positive *Impact Factor* and their subtrees as well from the element tree.

**Algorithm 1** The Main Procedure

**Input:**
1: The overall failed service calls in previous and latter intervals ($t_0$, $t_1$), $Fn(*, t_0)$, $Pn(*, t_1)$;
2: The overall service calls in previous and latter intervals ($t_0$, $t_1$), $Rn(*, t_0)$, $Rn(*, t_1)$;
3: The dataframe ($df$) of service calls within the two intervals ($t_0$, $t_1$), $df(t_0)$, $df(t_1)$;
**Output:**
4: Clues Set, $ClueSet$;
5: **function** MAIN( )
6: $rootNode \leftarrow new\ Node()$ //Create the root node.
7: CREATECHILDREN($rootNode, Fn(*, t_0), Fn(*, t_1), Rn(*, t_0), Rn(*, t_1), df(t_0), df(t_1)$) // Algorithm 2, init the first layer.
8: $ResultNodes \leftarrow []$
9: BFSLAYERS($rootNode.children, rootNode, ResultNodes, Fn(*, t_0), Fn(*, t_1), Rn(*, t_0), Rn(*, t_1), df(t_0), df(t_1)$) // Algorithm 2
10: // Calculate the WRS value of each node.
11: $sort(ResultNodes, key = WRS, type = ascending)$
12: $ClueSet \leftarrow ResultNodes.top(N)$
13: **return** $ClueSet$
14: **end function**

### 4.3 Overall algorithm

Overall, the major difference from *ImpAPTr* [1] exists in Algorithm 1, which is a breadth-first traversal algorithm on an element tree. Two major steps (i.e. Line 7 and Line 9) are detailed in Algorithm 2 and Algorithm 3, respectively. As shown in Algorithm 2, a new child node is created by modifying one of the $*$ (if exists) of the current element to a legal dimensional attribute value (step *"GenerateNewElements"*, *line 3 in Algorithm 2*). Fig. 2(b) provides an example to generate child nodes from the root node. Then we applied a breadth-first traversal algorithm to fetch candidate elements

**Algorithm 2** Grow Child Nodes

1: **function** CREATECHILDREN($currentNode, Fn(*, t_0), Fn(*, t_1), Rn(*, t_0), Rn(*, t_1), df(t_0), df(t_1)$)
2: $elements \leftarrow []$
3: $elements \leftarrow GenerateNewElements$
4: **for** each $e \in elements$ **do**
5: $node \leftarrow new\ Node()$
6: $Fn(e, t_0) \leftarrow AGG(df(t_0), e)$
7: $Rn(e, t_0) \leftarrow AGG(df(t_0), e)$
8: $Fn(e, t_1) \leftarrow AGG(df(t_1), e)$
9: $Rn(e, t_1) \leftarrow AGG(df(t_1), e)$
10: $SRSC(e, t_0), SRSC(*, t_0), SRSC(\neg e, t_0) \leftarrow$ // Equation 3
11: $SRSC(e, t_1), SRSC(*, t_1), SRSC(\neg e, t_1) \leftarrow$ // Equation 3
12: $IF(e) \leftarrow SRSC(*, t_1) - SRSC(\neg e, t_1)$
13: $Df(e) \leftarrow //Equation10$
14: // Set the values of the node.
15: $currentNode.children.add(node)$
16: $node.parent \leftarrow currentNode$
17: **if** $IF(e) >= 0$ **then**
18: $node.pruned \leftarrow TRUE$
19: **end if**
20: **end for**
21: **end function**

**Algorithm 3** Breadth-first Traversal

1: **function** BFSLAYERS($layerNodes, currentNode, ResultNodes, Fn(*, t_0), Fn(*, t_1), Rn(*, t_0), Rn(*, t_1), df(t_0), df(t_1)$)
2: // Delete the redundant nodes.
3: $nextLayer \leftarrow []$
4: **for** each $node \in layerNodes$ **do**
5: **if** $node.pruned$ != $TRUE$ **then**
6: CREATECHILDREN($node, Fn(*, t_0), Fn(*, t_1), Rn(*, t_0), Rn(*, t_1), Ndarray(t_0)$ , $Ndarray(t_1)$)
7: **end if**
8: **if** node is the rightmost **then**
9: **for** each $nod \in layerNodes$ **do**
10: **if** $nod.impactFactor < 0$ **then**
11: $ResultNodes.append(nod)$
12: **end if**
13: **end for**
14: **end if**
15: $nextLayer.append(node.children)$
16: **end for**
17: BFSLAYERS($nextLayer, currentNode, ResultNodes, Fn(*, t_0), Fn(*, t_1), Rn(*, t_0), Rn(*, t_1), df(t_0), df(t_1)$)
18: **end function**

which might be potential clues according to their *Ranking Score (RS)*. The generation of the element tree is along with the traversal with the strategies discussed in Section 4.2 to limit the size of the final element tree. Finally, as shown in Algorithm 1, we sort all the nodes by *WRS* ascendingly and generate the top $N$ possible clues.

## 5 EVALUATION

In this section, we elaborate the evaluation on the performance of *ImpAPTr+* to identify the valid clues associated with anomalies regarding the effectiveness and efficiency. The former mainly focuses on whether a valid clue can be identified and the accuracy. The latter focuses on the efficiency of the identification process, meaning the time cost to run the algorithm and identify a valid clue. Two different evaluations with different datasets and research purposes are then designed and executed. In general, the evaluation is to identify proper elements/clues which lead to the root causes contributing to service anomalies regarding $DSR_\Delta$ in a timely manner. To complete the evaluation, we first retrieve the raw data from the production environment and then run *ImpAPTr+*. Based on the resulted data, we perform confirmation and analysis to understand the performance of *ImpAPTr+* regarding effectiveness and efficiency. With this

fundamental understanding, we then design and implement a more thorough evaluation on *ImpAPTr+* using a fabricated dataset and compare the results with the other two methods, i.e. *R-Adtributor* and *Squeeze*.

To address the research objective, we establish two research questions for evaluation as follows.

RQ1. *Is ImpAPTr+ able to identify the valid clues (elements) of service anomalies regarding $DSR_\Delta$ at what accuracy leve?* In order to locate the valid clue, as the main objective of our research, we intend to test whether and to what degree this objective can be satisfied.

RQ2. *How long will it take to locate the valid clues of service anomalies regarding $DSR_\Delta$?* Timeliness is also crucial if we intend to apply *ImpAPTr+* in production environment. Therefore, RQ2 is to explore the time needed to identify useful clues through *ImpAPTr+*.

### 5.1 Evaluation with production dataset

With the raised RQs, our first evaluation is conducted with the dataset from the real production environment.

#### 5.1.1 Research process

As shown in the upper lane of Fig. 3, the research process consists of three major steps.

1) One researcher (*Researcher A*) retrieved monitoring data from the production environment to prepare the dataset for evaluation.
2) Another researcher (*Researcher B*) ran *ImpAPTr+* on the dataset created from the first step.
3) We analyzed the resulted clues by working with the experienced operations engineers in *Meituan*. By digging out potential root causes through the clues we obtained from the second step, we attempted to confirm the effectiveness of *ImpAPTr+*.

#### 5.1.2 Dataset preparation

Dataset A involves real cases from the production environment. The first week of production environment data from March 1st, 2020 has been retrieved through the CAT tool. Without any preference, we feed *ImpAPTr+* with the data from a serial of time intervals where *SRSC* is relatively high and changes gently since the major issue *ImpAPTr+* targeting is actually trivial anomalies. Otherwise, the previous *ImpAPTr* is capable enough to find the correct clues [1]. Meanwhile, to fully evaluate our method, we include all the seven dimensional attributes in dataset A. Therefore, one example record with JSON format is {`Network`: 4G, `Connect-type`: Type1, `Platform`: android, `ISP`: CMCC, `City`: SH, `App-source`: APP1, `App-version`: 10.1.0, `Code`: 200}.

#### 5.1.3 Execution

The experiments involved in this evaluation were run on a virtual server, which configured as Linux OS with Intel Core Processor (Skylake) @ 2095.074 MHz and 16GB Memory. The runtime is Python-3.8 development environment. As shown in Fig. 3, *Researcher A* prepared the dataset with separate JSON files, then *Researcher B* ran *ImpAPTr+* to figure out the clues. As the last step, researchers worked with the front-line operations staff to study the clues and try to dig out the root causes. However, this is a time-consuming task. We discussed a dozen cases, some of which are listed in TABLE 6.

#### 5.1.4 Result analysis

As mentioned before, the actual production environment data contains all the seven dimensional attributes, some of which contains thousands of valid values. As TABLE 6 presents, all the three cases show very high *SRSC* during the time periods which are considered 'healthy' status normally. Although there are considerable number of failed requests in these three cases, anomaly detection or root cause location methods will not be applied in most cases. However, *ImpAPTr+* managed to identify useful clues based on the quite 'healthy' data. The time to run *ImpAPTr+* for the three cases are 18.027, 43.2306, and 22.9687 seconds, respectively. The root causes or deeper clues for some of the clues have also been dug out. Take the software errors for example, which involves case 1 and case 3. The *SV1* is an issue about certain special characters in the requests from Android devices being parsed as wrong-format data and further incurring failed response. Meanwhile, the *SV2* is an issue regarding multithreaded concurrency. With relatively high QPS (Queries Per Second), a recently updated component (through connection Type8) does not set lock control between threads, causing occasional deadlocks for some threads. Interestingly, the top 2 clues in case 3 are pointing to one same root cause, which is not rare in the clues generated by *ImpAPTr+*.

In general, *ImpAPTr+* has successfully located the valid clues leading to the root causes of the service anomalies, which to a certain degree confirms the effectiveness of the improved method (RQ1). Meanwhile, the time for locating is normally less than one minute for time period less than one hour, which is timely and efficient (RQ2).

### 5.2 Evaluation with simulated dataset

The evaluation with the simulated dataset intents to portray the performance of *ImpAPTr+* extensively. To achieve this goal, we planted known anomalies to increase the anomaly density.

#### 5.2.1 Research process

As shown in the lower lane of Fig. 3, the research process of the evaluation on the simulated dataset also consists of three major steps, except the first step is more complicated than the evaluation with real production environment dataset.

1) *Researcher A* prepared the simulated dataset based on the monitoring data from the production environment (cf. Section 5.2.2 for detail).
2) *Researcher B* ran *ImpAPTr+* on the dataset created from the first step.
3) we analyzed the resulted clues to evaluate the performance of *ImpAPTr+* in terms of effectiveness and efficiency.

TABLE 6: Identification results from production environment dataset

| **Date** | 3.2 | **Time period** | 20:00∼20:40 | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Time interval** | T1 | T2 | T3 | T4 | T5 | T6 | | |
| **#Records** | 1485880 | 1492475 | 1504285 | 1476165 | 1434970 | 1434559 | **Clue Identified in 18.0273s** | **Confirmation Results** |
| **SRSC** | 99.779% | 99.773% | 99.774% | 99.767% | 99.768% | 99.756% | | |
| **#Failed requests** | NA [a] | NA[a] | 358 | 377 | 360 | 382 | 1st Clue (Type0, Shanghai) | CSD[c] |
| **#Failed requests** | 204 | 298 | 291 | 330 | 332 | 333 | 2nd Clue (Type0, Android, SH) | SV1[d] |
| **#Failed requests** | NA[a] | 1187 | 1191 | 1186 | 1199 | 1250 | 3rd Clue (WIFI, Type0, Android) | NA[b] |
| **Date** | 3.3 | **Time period** | 12:15∼13:15 | | | | | |
| **Time interval** | T1 | T2 | T3 | T4 | T5 | T6 | | |
| **#Records** | 2018526 | 1957376 | 1927742 | 1880492 | 1840619 | 1797344 | **Clue Identified in 43.2306s** | **Confirmation Results** |
| **SRSC** | 99.795% | 99.784% | 99.772% | 99.771% | 99.761% | 99.754% | | |
| **#Failed requests** | NA[a] | 326 | 443 | 473 | 372 | 479 | 1st Clue (IOS, ISP:other) | JSC[e] |
| **#Failed requests** | NA[a] | 310 | 514 | 556 | 367 | 509 | 2nd Clue (ISP:other, Guangdong) | NA[b] |
| **#Failed requests** | NA[a] | NA [a] | NA[a] | 247 | 394 | 162 | 3rd Clue (Unicom, Shanxi) | NA[b] |
| **Date** | 3.4 | **Time period** | 12:00∼12:40 | | | | | |
| **Time interval** | T1 | T2 | T3 | T4 | T5 | T6 | | |
| **#Records** | 2529157 | 2430195 | 2372317 | 2236694 | 2150281 | 2039741 | **Clue Identified in 22.9687s** | **Confirmation Results** |
| **SRSC** | 99.812% | 99.801% | 99.800% | 99.791% | 99.784% | 99.776% | | |
| **#Failed requests** | NA[a] | 430 | 339 | 484 | 413 | 409 | 1st Clue (WIFI, Type8) | SV2[f] |
| **#Failed requests** | NA[a] | NA | 399 | 481 | 413 | 408 | 2nd Clue (WIFI, Type8, IOS) | SV2[g] |
| **#Failed requests** | NA[a] | NA[a] | NA[a] | 258 | 259 | 256 | 3rd Clue (Type0, Android, Telecom) | NA[b] |

[a] No record has been identified to be related to the clue at the same line.
[b] No root cause or valuable information about the root cause has been dug out through the clue at the same line.
[c,e] ISP errors, however, due to privacy policy, we omit the details.
[d,f,g] Software errors, see Section 5.1.4 for the detail.

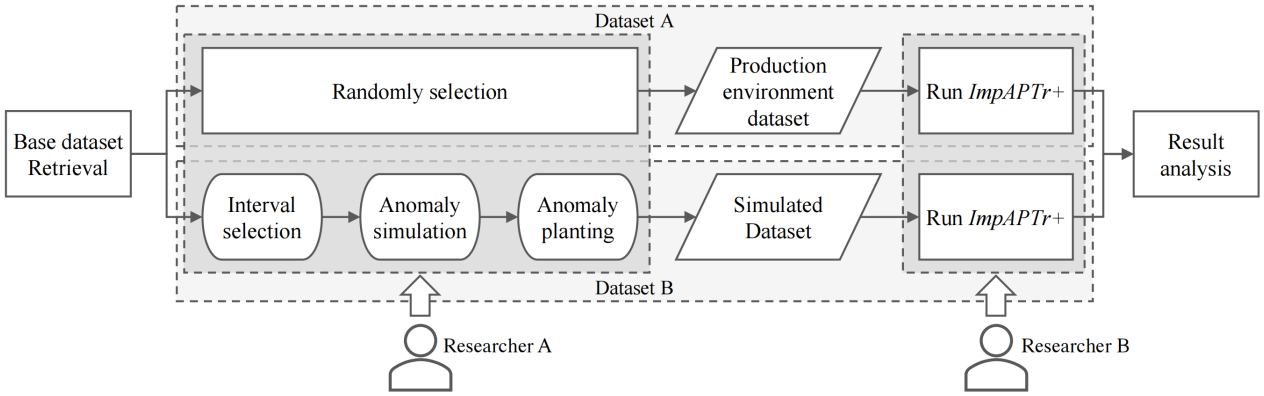



Fig. 3: The process of the research

### 5.2.2 Dataset preparation

The results in the previous evaluation have revealed that to locate a clue pertinent to one anomaly, the number of continuous time intervals is normally less than 6 (a.k.a. 30 minutes). Meanwhile, no evidence has been observed to imply obvious regularities of the occurrence of failed service requests across continuous time intervals. With these observations, together with the recommendation suggested in [58] on minimal criterion on the number of random tests, we simulated more than 3000 sets of data (each contains 30-minute information on service requests) to perform the evaluation. However, to construct a simulation dataset is not easy, which needs subtly design. We took several steps to construct the simulation dataset as follows.

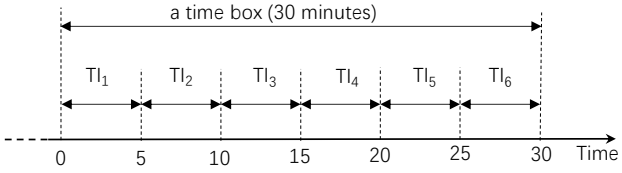



Fig. 4: A time box of 30 minutes with 6 intervals

**Step 1:** **Base dataset retrieval.** The base dataset is from the real production environment in *Meituan*. The APM system stored the monitoring data in a database, from which we retrieved the monitoring data of the *DP* system from January 1st to January 31st, 2020. To be specific, we retrieved the service calls to a hot service named "shop". For each day, there are around 20 to 70 million service calls to this service, spreading over 288 time intervals. We selected the data from 10:00∼20:00, resulting 120 time intervals each day. The result is presented in TABLE 7. Take January 1st as an example, we have 120 time intervals, and the range of the number of requests for each time interval is also listed, i.e. 385971 on average and the 25% and 75% region ranges from 358566 to 436524.

**Step 2:** **Time box selection.** A time box contains 30 minutes which is formed by 6 continuous time intervals, as shown in Fig. 4. Without any preference, one researcher constructed the time boxes from 10:00∼20:00 each day. The basic idea is to use the first 30 minutes (10:00∼10:30) as the first time box, and then using 5 minutes (one time interval) as a time window, pan to the right to generate the rest time boxes. As a result, there are 115 time boxes

TABLE 7: Daily experimental data in January, 2020.

| Date | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| (25%).Requests | 358566 | 230472 | 246805 | 313465 | 277496 | 211669 | 223854 | 226433 | 234644 | 250803 | |
| Avg.Requests | 385971 | 257245 | 284617 | 346418 | 298673 | 237122 | 251089 | 256707 | 265137 | 290421 | |
| (75%).Requests | 436524 | 283778 | 326333 | 375109 | 319594 | 265420 | 277766 | 286264 | 295176 | 329153 | |
| Date | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | |
| (25%).Requests | 320829 | 302917 | 238110 | 248434 | 250162 | 250294 | 258066 | 317481 | 282424 | 262486 | |
| Avg.Requests | 353301 | 328140 | 266781 | 279017 | 281766 | 282945 | 299844 | 345413 | 315582 | 298319 | |
| (75%).Requests | 379275 | 359309 | 297257 | 308226 | 312215 | 313221 | 337928 | 371306 | 341800 | 324290 | |
| Date | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| (25%).Requests | 231468 | 200177 | 149416 | 82124 | 90211 | 80317 | 78230 | 75513 | 69826 | 65728 | 60893 |
| Avg.Requests | 261975 | 220911 | 157788 | 89561 | 95638 | 84992 | 83399 | 79699 | 73639 | 69884 | 64487 |
| (75%).Requests | 289390 | 238725 | 171266 | 96196 | 103091 | 91632 | 88755 | 86055 | 79292 | 74824 | 69363 |

* The number in this table is slightly different from our previous study [1] due to different time scopes.

daily, which were stored into a list $L_t$ for further simulation.

**Step 3:** **Anomaly simulation.** To simulate an anomaly, we need to simulate a combination of dimensional attributes ($e$) and a $DSR_\Delta$ caused by $e$ at the same time. **Randomness** is essential to simulate both $e$ and $DSR_\Delta$ .

*Step 3.1*: *Element $e$ simulation.* To simulate a combination of dimensional attributes ($e$), we first construct an element tree (as shown in Fig. 2(a)) based on the time box determined in the previous step. In order to mimic seemly 'healthy' status, from the root we calculate $Rn(e_i, T)$ (i.e. the total number of requests within time interval $T$ under the element $e_i$) for each node (element $e_i$) to find all the $e_i$ with its $Rn(e_i, T)$ ranging from $0.01\%$ to $0.03\%$ of the number of all the service calls within a certain time interval for all the 6 time intervals contained in the time box. These elements were stored in a candidate list $L_e$ and from this list, we randomly selected one as the simulated element, which was used to plant anomalies in all the 6 time intervals.

*Step 3.2*: *DSR degree simulation.* In general, a $DSR_\Delta$ anomaly in this study is designed as the *SRSC* dropping less than $0.05\%$ in any two adjacent time intervals within the time box. Since we did not intend to change the base dataset too much, we randomly set the range for $DSR_\Delta$ from $0.01\%$ to $0.03\%$ for each time interval in the selected time box.

**Step 4:** **Anomaly planting.** To plant an anomaly in any time interval as discussed in the above steps, we need to simultaneously consider two adjacent time intervals, including the nearest time interval (i.e. $T_0$ in Fig. 4). An object $DSR_0$ is randomly selected from $0.01\%$ to $0.03\%$ (Step 3.2). For a given $e$, the upper limit of the number of successful requests which need to be changed from successful requests to failed requests is thus determined by Equation 5 to create the $DSR_0$, meaning that a less than $DSR_0$ anomaly may happen if there are not enough successful service requests. Unlike [1], we do not need to reselect a new element. Therefore, the last step of anomaly planting is to randomly select these service calls constrained by $e$ and change its status from "successful request" to "failed request".

$$Rn(T_2) * (SRSC(T_2) - SRSC(T_1) + DSR_0) \quad (5)$$

where $T_1$ and $T_2$ denote two adjacent time intervals in a time box, $Rn(T_i)$ represents the number of requests within time interval $T_i$.

#### 5.2.3 Execution

We used the same configuration described in Section 5.1.3 to run *ImpAPTr+* on the simulated dataset. Due to limited memory, we stored the simulated data in separated JSON files. To be specific, as described in Section 5.2.2, we selected all suitable time boxes according to *Step 2*, then for each time box, one researcher tried 115 times (as a balance between the number of random simulations and computation resources) to simulate $e$ (*Step 3.1*) and plant an anomaly (*Step 4*) and stored into separated JSON files. As a result, we have 3565 files, corresponding to 3565 time boxes with various planted anomalies. We then transferred all the JSON files to another researcher who ran *ImpAPTr+* and try to locate the valid clues pertinent to the planted anomalies.

#### 5.2.4 Result analysis

To answer the RQs raised in Section 5, we need to analyze the resulted data from various aspects.

A. Accuracy: Apparently, the first and foremost criterion to evaluate the performance of *ImpAPTr+* should be the accuracy of locating the valid clues pertinent to the planted anomalies. Therefore, we define the accuracy as the percentage of successfully identified valid clues in 3565 time boxes, i.e. $accuracy = SI/3565$ where *SI* denotes the number of successful locating to valid clues. However, the preliminary trials imply that it was normally impossible to locate the exact one, we then loosened the criterion of *SI* to the top 3, 5 and 10 candidate clues. Take top 3 for example, if the actual $e$ exists in the top 3 clues after running *ImpAPTr+*, then we deem it a successful locating, otherwise a failed one. Top 5 and 10 apply similar rules to determine *SI*. Fig. 5 depicts the clue identification accuracy for the three methods. Apparently, *ImpAPTr+* outperforms the other two methods in all the three different criteria (i.e. top 3, 5 and 10) of accuracy. Take top 3 as an example, the accuracy of *ImpAPTr+* is $81.43\%$, indicating that out of 3565 simulations, *ImpAPTr+* can locate the correct clues 2902 times.

B. Efficiency: Due to the difficulty to identify clues associated with trivial anomalies as designed in the fabricated dataset, none of the three methods can correctly identify a clue all the time, we evaluated the efficiency of the three methods from three different aspects. The first is about the time required to analyze the monitoring data from

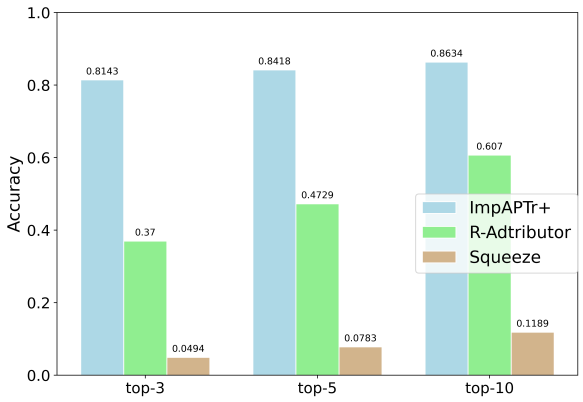



Fig. 5: Top 3, 5 and 10 accuracy for three methods

one time interval (5 minutes). The second aspect is thus the time required to identify a clue correctly. The third is the capability to identify a clue correctly within fewer time intervals.

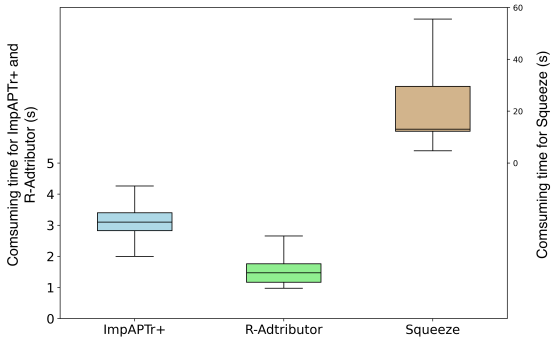



Fig. 6: Time required to analyze data for one time interval by three methods (in seconds)

TABLE 8: Time on processing data for one time interval per method

| | ImpAPTr+ | R-Adtributor | Squeeze |
|---|---|---|---|
| Max | 4.26 | 2.66 | 55.58 |
| 75% quartile | 3.4 | 1.76 | 29.59 |
| Avg | 3.12 | 1.5 | 20.89 |
| 25% quartile | 2.83 | 1.17 | 12.25 |
| Min | 1.79 | 0.98 | 4.73 |

B.1 Time to analyze the data from one time interval: As shown in Fig. 6 and TABLE 8, the three methods perform differently in terms of the time required to analyze the monitoring data coming from one time interval. *R-Adtributor* performs the best, which is able to complete the analysis from 0.98∼2.66 seconds, on average 1.5 seconds. *ImpAPTr+* requires a comparable period of time to complete the similar analysis, ranging from 1.79 seconds to 4.26 seconds, on average 3.12 seconds. While both the methods can perform the analysis in a matter of a few seconds, *Squeeze* may need tens of seconds to perform the similar analysis. Another

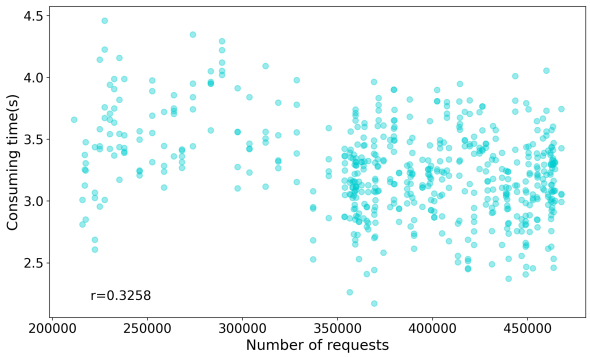



Fig. 7: Correlation between the time required to analyze the data from one time interval and the number of requests contained in the time interval, using *ImpAPTr+*

interesting finding is that the time required to perform analysis seems not related to the number of requests within a certain time interval. The correlation coefficient $r$ between these two equals $0.3258$, as shown in Fig. 7, which implies that the time required to analyze data from one time interval by *ImpAPTr+* depends on other factors. Nevertheless, given the characteristics presented in Fig. 7, time intervals with more service calls tend to cost less time on the analysis.

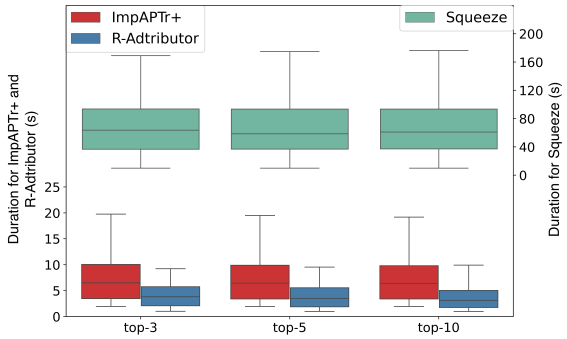



Fig. 8: Time required to identify a correct clue within top 3, 5 and 10 candidates for three methods

TABLE 9: The time to identify a correct clue within top 3, 5 and 10 candidates for three methods (in seconds)

| | ImpAPTr+ | R-Adtributor | Squeeze |
|---|---|---|---|
| Max (top 3/5/10) | 19.96/19.63/19.42 | 11.26/11.08/9.99 | 179.1/178.7/178.2 |
| 75% quartile (top 3/5/10) | 10.06/9.9/9.81 | 5.76/5.57/5.06 | 93.7/93.62/93.62 |
| Avg (top 3/5/10) | 7.47/7.37/7.3 | 4.08/3.88/3.57 | 65.66/65.93/65.87 |
| 25% quartile (top 3/5/10) | 3.46/3.41/3.4 | 2.09/1.89/1.77 | 36.76/36.9/37.23 |
| Min (top 3/5/10) | 1.97/1.97/1.97 | 1.03/1.01/1.01 | 9.98/9.98/9/98 |

B.2 Time to identify a clue correctly: The identification of a clue requires analysis across several time intervals by all the three methods. Fig. 8 and TABLE 9 present the time required to identify a correct clue using different methods with different criteria. Take the top 3 identification for example, apparently, *R-Adtributor* requires the least time to perform the clue identification, which ranges from 1.03 to 11.26 seconds, on average 4.08 seconds. *ImpAPTr+* requires a

little bit extra time to find a correct clue, which ranges from 1.97 to 19.96 seconds, on average 4.08 seconds. However, the time required to perform the similar identification by *Squeeze* varies from a few seconds to nearly 180 seconds, significantly much longer than that for its two competitors. In general, *R-Adtributor* and *ImpAPTr+* require comparable time, which is in a matter of several seconds to identify a clue correctly. Nevertheless, *Squeeze* requires almost 10 times longer to perform similar analysis.

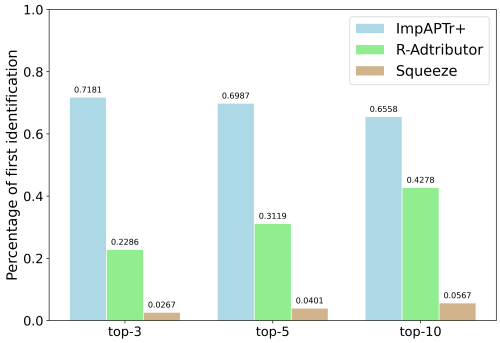



Fig. 9: Winning rate to identify a correct clue using the least time intervals

B.3 Timing to identify a clue correctly: As shown in Fig. 4, a time box contains 6 time intervals. Basically, all the three methods need multiple time intervals to identify a correct clue. Apparently, the fewer time intervals needed the more feasible to deploy the method to perform a near real-time clue identification. To compare the performance of a certain method in terms of timing, we measured how many times that it took the fewest time intervals to locate a correct clue. The results are presented in Fig. 9. Apparently, *ImpAPTr+* performs the best among all the methods with all the top $N$ criteria. Take top 3 as an example, the winning rate (i.e. the percentage of the times that one method used the smallest number of time intervals to identify a valid clue correctly) of *ImpAPTr+* is $71.81\%$, meaning that *ImpAPTr+* uses the fewest time intervals in $2560$ out of $3565$ time boxes to identify a clue correctly. One noteworthy point is that different methods may identify a correct clue in an exactly same time interval. In this case, both the methods are equally the winners regarding the timing. An interesting phenomenon, as shown in Fig. 9, is that with the looser criteria to determine a correct identification (i.e. top 3, 5 and 10), the winning rate of *ImpAPTr+* decreases, whilst the winning rate of the other two methods increase. This phenomenon is because looser criteria helps to include more candidate clues, which may imply that *ImpAPTr+* is more likely to locate a clue precisely.

In general, *ImpAPTr+* performs the best in terms of accuracy. In most cases, we can expect an accuracy over $80\%$ (RQ1), much higher than the other two methods. Meanwhile, although *ImpAPTr+* is not the fastest method to identify a correct clue, it only needs a matter of seconds to find a proper clue (RQ2). Last but not least, in most cases, *ImpAPTr+* requires the fewest time intervals to locate a correct clue, which shows the potential to support a near real-time monitoring and clue locating to on-line services.

### 5.3 Threats to validity

We discuss several concerns related to the threats to validity of this evaluation.

Dataset: To evaluate *ImpAPTr+* and explore its performance, we simulated anomalies and randomly inserted them into the raw dataset retrieved from the real production environment. In this way, we increased the density of anomalies so that a relatively comprehensive evaluation would not take weeks or even months, imaging the huge effort to confirm anomalies in the real production environment. However, the actual anomalies and the related service calls may take a different way to take place in the production environment. To mitigate this issue, we applied random strategy in both clue formation and service request selection. Besides, with reference to the suggestion raised in [58], 3565 times of random tests were adopted in our study to evaluate the performance of *ImpAPTr+*. Moreover, two researchers independently prepared the dataset and evaluated *ImpAPTr+*. In this sense, this threat to validity could be mitigated to a fair degree.

No severe DSR in evaluation using dataset A: This study is motivated to free the constraint derived from a predefined threshold on $DSR_{\Delta}$. To mimic a seemingly 'healthy' status, we did not involve severe $DSR_{\Delta}$ in the evaluation using the dataset from production environment (i.e. dataset A). However, the algorithm behind *ImpAPTr+* does not rely on any degree of $DSR_{\Delta}$ , which implies that if *ImpAPTr+* can cope with trivial anomalies, it should be able to cope with severe $DSR_{\Delta}$s accordingly. Besides, the simulated dataset B covers continuous time boxes that contains dozens of severe $DSR_{\Delta}$s. The evaluation on the simulated dataset also confirms that *ImpAPTr+* is able to work with severe $DSR_{\Delta}$s.

Confirmation clues in dataset A evaluation: The evaluation on production environment dataset A generates several potential clues, for some of which researchers were not able to confirm the corresponding root causes. However, it might not be sufficient to justify invalid clues since the complexity of the service systems and relative high cost on mining the root causes. In this sense, the data characteristics (e.g., 30 minutes as the time box and no obvious regularity on the occurrence of the failed records, etc.) applied in the simulation may not be able to reflect all the situations, which raises the necessity for further investigation.

Combination of anomalies: In the simulated dataset, we did not compose combinations of anomalies to test *ImpAPTr+*, though the original motivation of this study is to address this issue existing in *ImpAPTr* [1]. On the one hand, the algorithm behind *ImpAPTr+* does not rely on any degree of $DSR_{\Delta}$ , which implies that *ImpAPTr+* treats anomalies one by one. In this sense, the concept of anomaly combination does not need to be considered when running *ImpAPTr+*. The identification results presented in TABLE 6 also imply that *ImpAPTr+* is able to find different clues pertinent to different root causes (a.k.a. combination of anomalies). On the other hand, before planting anomalies, the data in the original base time box has already contained various failed requests derived from different anomalies. Therefore, there are anomaly combinations existing in the simulated dataset. Based on the results in Section 5.2.4, obviously, *ImpAPTr+*

can work with anomaly combinations.

Distribution of failed requests in one time box: Since no obvious patterns have been observed, we randomly set the number of requests which need to be changed from successful requests to failed ones across the six time intervals in one time box to mimic one anomaly. However, the anomalies in real-world may take certain patterns or distributions to occur and to impact the status of service requests. Nevertheless, all the three methods evaluated in this study do not rely on such patterns to identify anomalies. Meanwhile, the 3565 random tests may also mitigate this validity risk.

Time interval: *SRSC* and other critical metrics elaborated in Section 3.3.1 are calculated on a basis of 5-minute interval in this study. Apparently, *SRSC* is dynamic and uncertain with different start point or length of time interval. In this sense, more research should be carried out to explore a suitable way to set a reasonable interval regarding both the start point and length.

The dynamic service behavior and environment: The time box applied in current *ImpAPTr+* is six time intervals (i.e. 30 minutes), which also implies that the service and the environment should keep relatively comparable in the time box. However, with highly variable and non-stationary operating conditions, *ImpAPTr+* may not achieve correct clues leading to valid anomalies. In practice, operations staff can determine a proper timing to run *ImpAPTr+*.

The APM system: This study has been conducted using one of the popular APM systems we listed in Section 1 for data collection, which may limit its external validity. However, we only focused on data analysis in this study. As long as the data such as the number of requests, the successful rate of service calls, the multiple dimensional attributes are available in most APM systems, *ImpAPTr+* is able to work with different APM systems.

Multiple clues existing simultaneously: In practice, there may exist multiple valid clues simultaneously. *ImpAPTr+* cannot handle this situation directly by locating all these clues at the same time. However, *ImpAPTr+* identifies the most important clue to an anomaly according to *Weighted Ranking Score*. Therefore, if the root causes for other valid clues keep impacting the *SRSC* to a certain service, it is possible that these clues could be located if their *Weighted Ranking Score* increases.

## 6 Discussion

For many popular on-line software services, although the frequency of severe $DSR_\Delta$ might not be very high, slight $DSR_\Delta$s might occur continuously, which may also lead to severe consequences if the service is used by a large number of users and if the root causes of these slight $DSR_\Delta$s are not addressed in time. Meanwhile, the proliferation of DevOps and microservices architecture naturally encourage service autonomy and distributed deployment and operations. As a result, instead of single dimensional attribution being pertinent to an anomaly, there usually exists a combination of multiple dimensional attributions relating to one anomaly. Moreover, a severe $DSR_\Delta$ being caused by several trivial anomalies is also not rare in practice. Such situation makes service monitoring even more difficult to be conducted in a timely manner. We proposed *ImpAPTr+* to identify the valid clues indicating the root causes of service anomalies, which normally consist of multiple dimensional attributes. We discuss several considerations in this section.

Balancing is important: In general, service monitoring and clue identification inevitably consume computation resources, e.g., the overhead of service monitoring and CPU, memory resources, etc. Therefore, it is critical to maintain a balance between the cost of service monitoring and the benefit derived from monitoring. To be specific, *ImpAPTr+* is able to cope with a trivial anomaly pertinent to a $DSR_\Delta$ far less than $0.05\%$. If the total number of users is not very large during off-peak periods or the $DSR_\Delta$ is very small, it might not be economic to run *ImpAPTr+*, given the fact that the algorithm is not sensitive to the number of requests. Another example is the top 3, 5 and 10 possible clues we listed in *ImpAPTr+*. The balance is between the usable clues and the accuracy of the identification algorithm. In most cases, it does not make sense to list top 10 clues to improve accuracy. In general, factors such as the business scenario, the expectation from the relevant stakeholders and data characteristics should be considered when choosing the balancing strategy.

Timing to run *ImpAPTr+*: For services with massive users, *ImpAPTr+* can be used to diagnose potential root causes after severe anomaly occurs (e.g., more than $0.05\%$ on $DSR_\Delta$) or other perceptible events pertinent to service failures since the algorithm behind *ImpAPTr+* does not rely on any degree of $DSR_\Delta$. Nevertheless, a more promising application scenario might be the maintenance of the healthiness of on-line services. As Fig. 1 indicates, services with massive users in *Meituan* cannot reach $100\%$ in terms of *SRSC*, which implies that there must be some anomalies at any time. Therefore, *ImpAPTr+* can be used as a precaution mechanism for services with massive users when the peak of requests comes. Operations staff can run *ImpAPTr+* proactively to find clues pertinent to anomalies before severe consequence occurs.

The value of efficiency for *ImpAPTr+*: *ImpAPTr+* is not designed to handle abrupt anomalies. Instead, it is driven by the urge to address the root causes of some trivial anomalies and keep the on-line services a near 100% healthy so as to further improve the user experience to the on-line services operated by *Meituan*. To achieve this, an approach to provide near real-time monitoring and anomaly clue identification is important, which requires *ImpAPTr+* to take less time and less data (i.e. fewer time intervals) to complete the diagnosis. In this sense, we deemed the efficiency of *ImpAPTr+* important and evaluated it from three aspects, i.e., (1) the time to analyze the data from one time interval, (2) the time to identify a clue correctly, and (3) the timing to identify a clue correctly.

Scale up service monitoring and clue identification: In this paper, we discuss $DSR_\Delta$ and clues pertinent to anomalies on a single service basis, which may encounter challenges when scaling up the method. However, since the service monitoring is performed by the APM system and the *ImpAPTr+* method is conducted by a separated computer, the scale-up should not be a tough job in most cases. One

possible strategy is to apply the "80–20" rule to perform the service monitoring and deploy *ImpAPTr+* on the busiest services. Moreover, the APM system captures the traces of all the service calls, which can be used to portray the topology of all the services and potentially help to optimize service monitoring and clues locating. Nevertheless, the balance between cost and benefit should be considered prior to deploying *ImpAPTr+* to monitor more services.

Synthesis of clues: Another noteworthy point is that to synthesize clues to identify certain patterns of anomalies or create more valuable clues based on original clues produced by *ImpAPTr+*. Without the constraints from $DSR_{\Delta}$, *ImpAPTr+* naturally creates many clues pointing to service anomalies, then if several clues contain the same value for a specific dimensional attribute, combing these clues may provide a more profound description about the potential root causes. Take the results in TABLE 6 for example, although some clues may not be able to expose root causes, the combination of the clues in the same time period does offer certain new insight to the status of the services. However, it may need experience and time to do better synthesis of clues.

## 7 Conclusion

For many on-line software systems with massive users, the healthiness of the software systems is critical to ensure continuous and reliable services. Therefore, it is important to identify and address anomalies in a timely manner so as to mitigate possible negative impacts on business. Among many indicators related to anomalies, *SRSC* and $DSR_{\Delta}$ to certain '*hot*' services easily draw attention from the business and operations staff, yet the challenges also exist. One is the complex reasons (i.e. combinations of multiple-dimensional attributes $\mathscr{S}$) behind a $DSR_{\Delta}$, the other is the small time slot available to find the $\mathscr{S}$, given that numerous users may be impacted if the root cause has not been addressed quickly. We proposed an algorithm, *ImpAPTr*, to tackle these challenges in [1]. However, *ImpAPTr* works with a predefined threshold (i.e. $0.05\%$ of $DSR_{\Delta}$), which encounters several issues in field adoption, e.g., the inability to cope with anomaly combinations, the difficulty to determine a proper threshold, etc. In this paper, we propose *ImpAPTr+* that no longer needs to specify the threshold on $DSR_{\Delta}$ in advance.

At this stage, *ImpAPTr+* shows great potential to strengthen the APM system regarding service monitoring and issue tracking. However, we also notice the complexity of the whole on-line system and the production environment as well when we implement this approach for field trial. To this end, we suggest two promising topics for future work.

(1) It might be valuable to design and implement a strategy to support a dynamic interval for *ImpAPTr+*, by which the dynamic time interval could be adjusted according to the number of users per time unit. For example, during peak hours, the interval should be shorter so as to save monitoring cost such as computing resources.
(2) The identification of a valid clue does not ensure that the corresponding root cause could be successfully revealed and addressed. While manual troubleshooting is a time-consuming task in most cases, it may be critical to evaluate the value of investigating every single clue, given that most clues may be related to trivial anomalies. For example, prediction models could be used to estimate the potential impacts of various anomalies in the near future and then prioritize the clues accordingly.

## Acknowledgments

This work is jointly supported by the National Key Research and Development Program of China (No.2019YFE0105500) and the Research Council of Norway (No.309494), the Meituan, the National Natural Science Foundation of China (No.62072227), the Key Research and Development Program of Jiangsu Province (No.BE2021002-2), the Technology Innovation Fund of Nanjing University, as well as the Intergovernmental Bilateral Innovation Project of Jiangsu Province (No.BZ2020017).

We would like to record our special appreciation to Ms. Chunan Chen, Mr. Tian Ren, and Mr. Jian Ouyang from Meituan for their assistance in data analysis, root cause identification, and confirmation.

## References

[1] G. Rong, H. Wang, Y. You, H. Zhang, J. Sun, D. Shao, and Y. Xu, "Locating the clues of declining success rate of service calls," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2020, pp. 335–345.
[2] S. R. Goniwada, "Observability," in *Cloud Native Architecture and Design*. Springer, 2022, pp. 661–676.
[3] C. C. Aggarwal, "An introduction to outlier analysis," in *Outlier analysis*. Springer, 2017, pp. 1–34.
[4] Z. Zhou and P. Tang, "Improving time series anomaly detection based on exponentially weighted moving average (EWMA) of season-trend model residuals," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2016, pp. 3414–3417.
[5] Q. Yu, L. Jibin, and L. Jiang, "An improved arima-based traffic anomaly detection algorithm for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 12, no. 1, p. 9653230, 2016.
[6] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, pp. 1–39, 2012.
[7] Z. He, P. Chen, X. Li, Y. Wang, G. Yu, C. Chen, X. Li, and Z. Zheng, "A spatiotemporal deep learning approach for unsupervised anomaly detection in cloud systems," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
[8] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at microsoft," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 3009–3017.
[9] X. Zhang, J. Kim, Q. Lin, K. Lim, S. O. Kanaujia, Y. Xu, K. Jamieson, A. Albarghouthi, S. Qin, M. J. Freedman *et al.*, "Cross-dataset time series anomaly detection for cloud systems," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, 2019, pp. 1063–1076.
[10] G. Yu, Z. Cai, S. Wang, H. Chen, F. Liu, and A. Liu, "Unsupervised online anomaly detection with parameter adaptation for kpi abrupt changes," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1294–1308, 2019.
[11] 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*, 2018, pp. 187–196.
[12] W. Chen, H. Xu, Z. Li, D. Pei, J. Chen, H. Qiao, Y. Feng, and Z. Wang, "Unsupervised anomaly detection for intricate kpis via adversarial training of vae," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1891–1899.

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

[14] S. Zhang, C. Zhao, Y. Sui, Y. Su, Y. Sun, Y. Zhang, D. Pei, and Y. Wang, "Robust kpi anomaly detection for large-scale software services with partial labels."

[15] R. Fonseca, M. J. Freedman, and G. Porter, "Experiences with tracing causality in networked services." *INM/WREN*, vol. 10, no. 10, 2010.

[16] R. Fonseca, G. Porter, R. H. Katz, and S. Shenker, "X-trace: A pervasive network tracing framework," in *4th Symposium on Networked Systems Design and Implementation (NSDI 2007), April 11-13, 2007, Cambridge, Massachusetts, USA, Proceedings*, 2007.

[17] J.-G. Lou, Q. Fu, Y. Wang, and J. Li, "Mining dependency in distributed systems through unstructured logs analysis," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 1, pp. 91–96, 2010.

[18] M. Chow, D. Meisner, J. Flinn, D. Peek, and T. F. Wenisch, "The mystery machine: End-to-end performance analysis of large-scale internet services," in *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, 2014, pp. 217–231.

[19] B. Agarwal, R. Bhagwan, T. Das, S. Eswaran, V. N. Padmanabhan, and G. M. Voelker, "Netprints: Diagnosing home network misconfigurations using shared knowledge." in *NSDI*, vol. 9, 2009, pp. 349–364.

[20] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[21] 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*, 2005, pp. 173–178.

[22] J.-P. Vasseur, M. Pickavet, and P. Demeester, *Network recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Elsevier, 2004.

[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. Sebos, J. Yates, A. Greenberg, and D. Rubenstein, "Effectiveness of shared risk link group auto-discovery in optical networks," in *Optical Fiber Communication Conference*. Optical Society of America, 2002, p. ThO5.

[25] 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*, 2016, pp. 1–9.

[26] X. Pan, J. Tan, S. Kavulya, R. Gandhi, and P. Narasimhan, "Ganesha: Blackbox diagnosis of mapreduce systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 3, pp. 8–13, 2010.

[27] K. Nagaraj, C. Killian, and J. Neville, "Structured comparative analysis of systems logs to diagnose performance problems," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*. USA: USENIX Association, 2012, p. 26.

[28] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl, "Detailed diagnosis in enterprise networks," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, 2009, pp. 243–254.

[29] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier, "Using magpie for request extraction and workload modelling." in *OSDI*, vol. 4, 2004, pp. 18–18.

[30] S. Zhang, Y. Liu, D. Pei, Y. Chen, X. Qu, S. Tao, and Z. Zang, "Rapid and robust impact assessment of software changes in large internet-based services," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, 2015, pp. 1–13.

[31] 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.

[32] 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.

[33] 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.

[34] M. Steinder and A. S. Sethi, "Increasing robustness of fault localization through analysis of lost, spurious, and positive symptoms," in *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, 2002, pp. 322–331 vol.1.

[35] I. Rish, M. Brodie, and S. Ma, "Efficient fault diagnosis using probing," in *AAAI Spring Symposium on Information Refinement and Revision for Decision Making*, 2002.

[36] 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.

[37] 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*, 2015, pp. 464–476.

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

[39] M. Persson and L. Rudenius, "Anomaly detection and fault localization an automated process for advertising systems," Master's thesis, 2018.

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

[41] 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.

[42] Q. Lin, J.-G. Lou, H. Zhang, and D. Zhang, "idice: problem identification for emerging issues," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 214–224.

[43] 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*, vol. 25, no. 6, pp. 3709–3722, 2017.

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

[45] A. Faraz, E. Jeffrey, G. Zihui, L. Alex X., W. Jia, and Y. He, "Detecting and localizing end-to-end performance degradation for cellular data services," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9.

[46] Y. Xu, Z. Musgrave, B. Noble, and M. Bailey, "Bobtail: Avoiding long tails in the cloud," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI) 13*, 2013, pp. 329–341.

[47] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2015, pp. 38–44.

[48] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, vol. 89. Presses universitaires de Louvain, 2015, pp. 89–94.

[49] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[50] G. Dong and J. Bailey, *Contrast data mining: concepts, algorithms, and applications*. CRC Press, 2012.

[51] F. Lin, K. Muzumdar, N. P. Laptev, M.-V. Curelea, S. Lee, and S. Sankar, "Fast dimensional analysis for root cause investigation in a large-scale service environment," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 4, no. 2, pp. 1–23, 2020.

[52] V. Murali, E. Yao, U. Mathur, and S. Chandra, "Scalable statistical root cause analysis on app telemetry," in *2021 IEEE/ACM 43rd In-*

*ternational Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2021, pp. 288–297.

[53] V. Jeyakumar, O. Madani, A. Parandeh, A. Kulshreshtha, W. Zeng, and N. Yadav, "Explainit!–a declarative root-cause analysis engine for time series data," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 333–348.

[54] T. Islam and D. Manivannan, "Predicting application failure in cloud: A machine learning approach," in *2017 IEEE International Conference on Cognitive Computing (ICCC)*. IEEE, 2017, pp. 24–31.

[55] H. Jayathilaka, C. Krintz, and R. Wolski, "Performance monitoring and root cause analysis for cloud-hosted web applications," in *Proceedings of the 26th International Conference on World Wide Web (WWW)*, 2017, pp. 469–478.

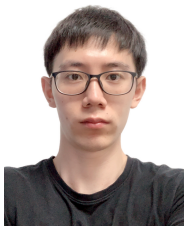
[56] M. Ma, Z. Yin, S. Zhang, S. Wang, C. Zheng, X. Jiang, H. Hu, C. Luo, Y. Li, N. Qiu *et al.*, "Diagnosing root causes of intermittent slow queries in cloud databases," *Proceedings of the VLDB Endowment*, vol. 13, no. 10, pp. 1176–1189, 2020.

[57] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, Jan 1991.

[58] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *2011 33rd International Conference on Software Engineering (ICSE)*, 2011, pp. 1–10.

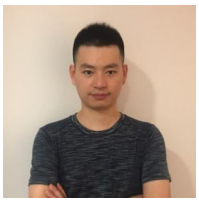
**Guoping Rong** is an associate researcher in Nanjing University. He received his BS, MS and Ph.D degree in Nanjing University. His research fields include BigData Technology, Empirical Software Engineering, Software Process and Project Management, DevOps, etc. He is a member of the IEEE Computer Society, the ACM(SIGSOFT), and the China Computer Federation(CCF).

**Hao Wang** is a postgraduate student in Nanjing University. His research fields include DevOps, AIOps, etc.



**Shenghui Gu** is a doctoral student in Nanjing University. His research interests are in software engineering, particularly in AIOps, software log analytics, DevOps, as well as empirical and evidence-based software engineering.

**Yangchen Xu** is a postgraduate student in Nanjing University. His research fields include DevOps, AIOps, etc.



**Jialin Sun** is a senior expert in Meituan, who leads the CAT team.

**Dong Shao** is an associate professor in Software Institute, Nanjing University. He is interested in Agile Software Development, High Tech Market, Management of Start-up, Software Engineering Education and general Software Engineering issues. He is a member of the IEEE Computer Society, the ACM and the CCF.

**He Zhang** is a Full Professor of Software Engineering and the Director of DevOps+ Research Laboratory at the Nanjing University, China, also a Principal Scientist with CSIRO, Australia. He is a Member of the IEEE Computer Society and the ACM (SIGSOFT), a Senior Member of China Computer Federation (CCF), and serves on the Steering Committees and Program Committees of a number of high quality international conferences in software engineering community.