Causal Inference-Based Root Cause Analysis for Online Service Systems with Intervention Recognition

Mingjie Li
Zeyan Li
Kanglin Yin
Tsinghua University
Beijing, China

Xiaohui Nie
Wenchi Zhang
Kaixin Sui
BizSeer
Beijing, China

Dan Pei*
Tsinghua University
Beijing, China

ABSTRACT

Fault diagnosis is critical in many domains, as faults may lead to safety threats or economic losses. In the field of online service systems, operators rely on enormous monitoring data to detect and mitigate failures. Quickly recognizing a small set of root cause indicators for the underlying fault can save much time for failure mitigation. In this paper, we formulate the root cause analysis problem as a new causal inference task named *intervention recognition*. We proposed a novel unsupervised causal inference-based method named *Causal Inference-based Root Cause Analysis* (CIRCA). The core idea is a sufficient condition for a monitoring variable to be a root cause indicator, *i.e.*, the change of probability distribution conditioned on the parents in the Causal Bayesian Network (CBN). Towards the application in online service systems, CIRCA constructs a graph among monitoring metrics based on the knowledge of system architecture and a set of causal assumptions. The simulation study illustrates the theoretical reliability of CIRCA. The performance on a real-world dataset further shows that CIRCA can improve the recall of the top-1 recommendation by 25% over the best baseline method.

CCS CONCEPTS

• **Software and its engineering** → *Software reliability*; • **Computing methodologies** → *Causal reasoning and diagnostics.*

KEYWORDS

root cause analysis, causal inference, intervention recognition, online service systems

ACM Reference Format:
Mingjie Li, Zeyan Li, Kanglin Yin, Xiaohui Nie, Wenchi Zhang, Kaixin Sui, and Dan Pei. 2022. Causal Inference-Based Root Cause Analysis for Online Service Systems with Intervention Recognition. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539041

*Dan Pei is the corresponding author. Email: peidan@tsinghua.edu.cn

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539041

1 INTRODUCTION

Fault diagnosis is critical in many domains, *e.g.*, machinery maintenance [28], petroleum refining [6], and cloud system operations [21, 30], which is an active research topic in the SIGKDD community. In this work, we focus on root cause analysis (RCA) in online service systems (OSS), such as social networks, online shopping, search engine, *etc.* We adopt the terminology in [19], denoting a *failure* as the undesired deviation in service delivery and a *fault* as the cause of the failure.

With the expansion of system scale and the rise of microservice applications, OSS are more and more complex. As a result, operators rely on monitoring data to understand what happens in the system [2]. Common monitoring data include metrics, semi-structural logs, and invocation traces. As the most widely available data, metrics are usually in the form of time series sampled at a constant frequency, *e.g.*, once per minute. Several metrics are the measures of the overall system health status, named the *service level indicator*s (SLI), *e.g.*, the average response time of an online service. Once an SLI violates the pre-defined service level objective (*i.e.*, a failure occurs), operators will mitigate the failure as soon as possible to prevent further damage. As a single fault may propagate in the system [9] with multiple metrics being abnormal during a failure (named *anomaly storm* [31]), RCA (recognizing a small set of *root cause indicators*) of the underlying fault can save much time for failure mitigation.

With the rising emphasis on explainability in many domains, *causal inference* [27] has attracted much attention in the literature. Though causal inference is promising, causal inference-based RCA is little studied, except Sage [8] with counterfactual analysis. In this paper, we novelly map a fault in OSS as an intervention [20] in causal inference. From this point of view, we name a new causal inference task as *intervention recognition* (IR), *i.e.*, finding the underlying intervention based on the observations (Definition 2.1). Hence, we formulate RCA in OSS as an IR task.

The first challenge of the new IR task is the lack of a solution. Though Sage [8] conducts RCA via counterfactual analysis, the design of Sage implies an implicit assumption, *i.e.*, there is no intervention to the system. Hence, Sage is not a solution to the IR task. Based on the definition of IR, we find that the probability distribution of an intervened variable changes conditioned on parents in the Causal Bayesian Network (CBN). This Intervention Recognition Criterion points out an explainable way to conduct RCA.

The second challenge is to obtain the CBN for causal inference in OSS. Many works have been done for *causal discovery* [10] from observational data. MicroHECL [15] and Sage [8] utilize the call graph in OSS, which operators are familiar with. However, these

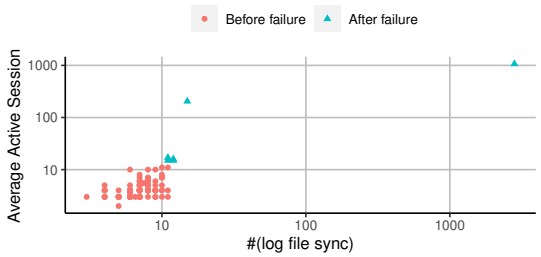


Figure 1: Joint distribution of the *Average Active Session* (an SLI of the Oracle database) and the number of *log file sync* waiting events within 2 hours. Each data point represents the two metrics' values at the same timestamp.

two works consider a few metrics, *e.g.*, the latency between services. We construct the CBN among metrics with the domain knowledge of system architecture, combined with a set of intuitive assumptions, handling more kinds of metrics than MicroHECL [15] and Sage [8].

Thirdly, observational knowledge is incomplete, indicating the difficulty of reaching interventional knowledge even with a perfect CBN. For example, Figure 1 shows the joint distribution of the *Average Active Session* (AAS) and the number of *log file sync* waiting events around a high AAS failure of an Oracle database instance. Observed data before the failure are in the bottom-left corner of the figure. Hence, how AAS normally distributes is missing when "#(log file sync)" is larger than 1,000, where the data after the failure distribute. The lack of overlap between the two distributions around the failure blocks recognizing intervention, if any, in AAS. To address this challenge, we transform distribution comparison as point-wise hypothesis testing via the regression technique. Moreover, a descendant adjustment technique is proposed to alleviate the bias introduced by a poor understanding of the system's normal status in the hypothesis testing.

We implement the proposed Causal Inference-based Root Cause Analysis (CIRCA). CIRCA outperforms baseline methods in our simulation study, illustrating its theoretical reliability. We further evaluate CIRCA with a real-world dataset. CIRCA improves the recall of the top-1 recommendation by 25% over the best baseline method, which shows the practical potential of our approach. The contributions of this work are summarized as follows.

- For the first time in the literature, we formulate the RCA problem in OSS as a new causal inference task named *intervention recognition* (Definition 2.1). Utilizing the advance of causal inference, we find a practical criterion to locate the root cause (Theorem 3.4).
- We propose Causal Inference-based Root Cause Analysis (CIRCA) for OSS. We propose a practical guideline to construct the CBN with the knowledge of system architecture. Two more techniques, namely regression-based hypothesis testing and descendant adjustment, are proposed to infer root cause metrics in the graph.
- CIRCA is evaluated with both simulation and real-world datasets. The simulation study illustrates CIRCA's theoretical reliability, while the real-world dataset shows CIRCA's practical value over baseline methods.

2 PROBLEM FORMULATION

2.1 Preliminary

Notation. An upper case letter (*e.g.*, X) refers to a variable (metric), while a lower case letter (*e.g.*, x) represents an *assignment* of the corresponding variable. By *assignment*, we mean one of the possible values. To distinguish variables (values) at different times in a time series, the timestamp will be put on the letter as a superscript. For example, denote AAS as an upper case letter Y, and $y^{(t)}$ refers to the value of AAS at time t. Denote the value range of Y as $Val(Y)$, then we have $y^{(t)} \in Val(Y) = \{0\} \cup \mathbb{R}^+$ for the non-negative numeric AAS. A boldfaced letter means a set of elements (variables or values), *e.g.*, we denote all the metrics as $\mathbf{V}$ while $\mathbf{v}$ is an assignment of $\mathbf{V}$.

The Ladder of Causation. We formulate the problem with Judea Pearl's "Ladder of Causation" [1]. The first layer of the causal ladder encodes the observational knowledge $\mathcal{L}_1(\mathbf{V}) = P(\mathbf{V})$, where $P(\mathbf{V})$ is a joint probability distribution. Meanwhile, the second layer encodes the interventional knowledge $\mathcal{L}_2(\mathbf{m}) = P_{\mathbf{m}}$, where $P_{\mathbf{m}}(\mathbf{V}) = P(\mathbf{V} \mid do(\mathbf{m}))$ and $\mathbf{M} \subseteq \mathbf{V}$. The *do-operator* $do(\mathbf{m})$ means fixing variables $\mathbf{M}$ to the given values $\mathbf{m}$, also called an *intervention* [20]. So that $P(\mathbf{V} \mid do(\mathbf{m}))$ indicates the probability distribution over $\mathbf{V}$ under the intervention to $\mathbf{M}$. Finally, the third layer encodes the counterfactual knowledge, reasoning about what if another situation happened in the past. For example, it requires the counterfactual knowledge to predict the latency with sufficient computing resources when high latency and full CPU usage are observed. The hierarchy of the causal ladder *almost never* collapses (named CHT, Causal Hierarchy Theorem [1]). If we want to answer the question at Layer i, we need knowledge at Layer i or higher [1].

Structural Causal Model (SCM). We model the relations among metrics via the structural causal model [20]. An SCM contains a set of structural equations shown in Eq. (1), where $V_i \in \mathbf{V}$ and $\mathbf{Pa}(V_i) \subseteq \mathbf{V}$. Eq. (1) contains two kinds of parameters: 1) assignments of observed variables $\mathbf{Pa}(V_i)$, named parents (direct causes) of V_i, and 2) assignments of unobserved variables $\mathbf{U}_i$, where $\mathbf{U}_i \cap \mathbf{V} = \emptyset$.

$$v_i = f_i(\mathbf{pa}(V_i), \mathbf{u}_i) \tag{1}$$

Denote the graph encoded by the SCM as $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{E} = \{V_j \to V_i \mid V_j \in \mathbf{Pa}(V_i)\}$ is the set of directed edges. In contrast to $\mathbf{Pa}$, $\mathbf{Ch}(V_i) = \{V_j \mid V_i \in \mathbf{Pa}(V_j)\}$ represents the children of V_i. This work rests on the following assumptions.

DAG $\mathcal{G}$ is a directed acyclic graph (DAG) [20], following related works in OSS [4, 8, 25].

Markovian "The exogenous parent sets $\mathbf{U}_i, \mathbf{U}_j$ are independent whenever $i \neq j$" [1], *i.e.*, $(\forall i \neq j)\mathbf{U}_i \perp\!\!\!\perp \mathbf{U}_j$, where $\perp\!\!\!\perp$ means independent.

Faithfulness [20] Any intervention makes an observable change, *i.e.*, $P(V_i \mid \mathbf{pa}(V_i), do(v_i)) \neq P(V_i \mid \mathbf{pa}(V_i))$.

Under the DAG assumption and the Markovian assumption, $\mathcal{G}$ can be taken as a CBN [1].

2.2 Root Cause Analysis and Causal Inference

We set up a concept mapping between the RCA problem and causal inference.

- A fault in OSS is mapped to an unexpected intervention;
- Fault-free data come from the observational distribution;

- Faulty data come from an interventional distribution.

Based on the mapping above, we define a new causal inference task as *intervention recognition* (Definition 2.1). We formulate RCA discussed in this work as an intervention recognition task in OSS.

Definition 2.1 (Intervention Recognition, IR). For a given SCM $\mathcal{M}$, let $\mathcal{L}_1$ be the observational distribution of $\mathcal{M}$ and $P_m = P(\mathbf{V} \mid do(\mathbf{m}))$ be the interventional distribution of a certain intervention $do(\mathbf{m})$. *Intervention recognition* is to find $\mathbf{m}$ based on $\mathcal{L}_1$ and P_m.

Definition 2.2 (Root Cause). The *root cause* is the intervened variables ($\mathbf{M}$). Each element of $\mathbf{M}$ is named a *root cause variable*.[1]

3 INTERVENTION RECOGNITION CRITERION

We argue that IR shall be positioned at the second layer in the ladder of causation, as shown in Theorem 3.1. The proof of Theorem 3.1 is provided in Appendix A. The key to the proof is that IR is the inverse mapping of $\mathcal{L}_2$ under the adopted assumptions. Combining Theorem 3.1 with CHT [1], we further obtain Corollary 3.2 and 3.3.

Theorem 3.1. *For a given SCM $\mathcal{M}$ with a CBN $\mathcal{G}$, the knowledge of IR for $\mathcal{M}$ is equivalent to $\mathcal{L}_2$ under the Faithfulness assumption.*

Corollary 3.2. *We need the knowledge at Layer 2 (interventional) to conduct IR.*

Corollary 3.3. *The knowledge at Layer 3 (counterfactual) is not necessary to conduct IR.*

Hence, we propose to take full advantage of the CBN, as the CBN is a known bridge between observational data and interventional knowledge [1]. We argue that Theorem 3.4 is a necessary and sufficient condition for a variable to be intervened. The proof of Theorem 3.4 is provided in Appendix B. Based on our concept mapping between RCA and causal inference, the Intervention Recognition Criterion is also a criterion to find root cause indicators.

Theorem 3.4 (Intervention Recognition Criterion). *Let $\mathcal{G}$ be a CBN and $\mathbf{Pa}(V_i)$ be the parents of V_i in $\mathcal{G}$. Under the Faithfulness assumption, V_i is intervened iff V_i no longer follows the distribution defined by $\mathbf{pa}(V_i)$, i.e.,*

$$V_i \in \mathbf{M} \Leftrightarrow P_{\mathbf{m}}(V_i \mid \mathbf{pa}(V_i)) \neq \mathcal{L}_1(V_i \mid \mathbf{pa}(V_i))$$

4 CAUSAL INFERENCE-BASED ROOT CAUSE ANALYSIS

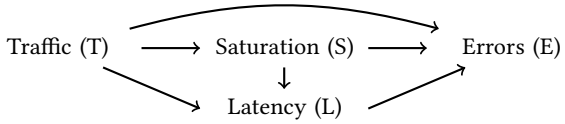
In this section, we propose a novel method named *CIRCA*. We first present a structural way to determine the parents $\mathbf{Pa}(V_i)$ for each metric V_i based on system architecture. CIRCA adopts *regression-based hypothesis testing* (RHT) to deal with the incomplete distribution of faulty data. To address the challenge of the incomplete distribution of fault-free data, CIRCA adjusts the anomaly score for suspicious metrics based on their descendants in the CBN.

4.1 Structural Graph Construction

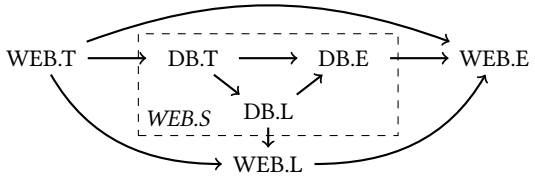
We propose the *structural graph* (SG) as the CBN for OSS. SG combines the system architecture knowledge with a set of assumptions, which may not suit domains other than OSS. We first classify monitoring metrics into four dimensions, named *meta metric*s. Several causal assumptions among those four kinds of meta metrics provide the building blocks of an SG. We further extend the system with the architecture of components to construct a graph at the meta metric level, named a *skeleton*. Finally, we plug monitoring metrics into the corresponding meta metric to obtain the SG. Algorithm 1 summarizes the overall procedure.

4.1.1 Meta Metrics. In general, a service takes *input* and produces *output*. Each request lasts for some *time* and consumes some *resources*. We take those dimensions as four *meta metrics* of a service, named after the four golden signals in site reliability engineering [2]. *Traffic*, *Errors*, and *Latency* measure the distribution of input, output, and processing time, respectively. We classify other monitoring metrics as resource consumption, denoted as *Saturation*.

We assign directions for the relations among these four meta metrics in Figure 2(a). As the start of a request, *Traffic* is assumed to be the cause of all other three meta metrics, while *Errors* (the end of a request) are taken as the effect of others. The edge from *Saturation* to *Latency* encodes our preference on the former, as resource consumption is one of the common considerations for large latency in OSS [8].



(a) Causal assumptions within a service



(b) The skeleton of one web service (WEB) with its dependent database (DB). We plug DB's meta metrics into the *Saturation* of WEB.

Figure 2: Causal assumptions among meta metrics

4.1.2 Skeleton with Architecture Extension. A complex OSS system will invoke multiple services to process one single request. Meanwhile, there will be multiple components for monolithic OSS. Based on the architecture knowledge encoded in the call graph, we construct the *skeleton* among meta metrics of the system and all its dependent services. For a web service (WEB) and its database (DB) shown in Figure 2(b), we take DB as a resource of WEB. The part of WEB's Saturation that measures DB will be extended into DB's meta metrics, which inherit the relations between WEB's Saturation and other meta metrics of WEB. The extension will be applied to each service in the call graph. In summary, we introduce three more causal assumptions between a service and its dependent ones.

- The caller's *Traffic* influences the callees' *Traffic*;
- The callees' *Latency* contributes to the caller's *Latency*;
- The caller's output is calculated based on the callees' output.

[1] We also use *root cause indicator* and *root cause metric* according to the context.

4.1.3 Monitoring Metric Plugging-in. Finally, we plug monitoring metrics in meta metrics to obtain the SG. A mapping is required to describe which dimension of which service each monitoring metric measures. There can be some meta metrics that do not have any monitoring metrics. For example, the common measurement for memory is just usage (*Traffic*), while the speed (*Latency*) is unavailable. Moreover, one monitoring metric can be derived from multiple meta metrics. For example, *DB access per request* is calculated by the Traffic of both a web service and a database.

Algorithm 1 describes the plugging-in process after skeleton construction. SG links monitoring metrics from one meta metric to its children (Line 15). Monitoring metrics that are derived from multiple meta metrics may introduce self-loop. To avoid such cycles, the monitoring metric for the last meta metric in topological order will be taken as the common effect of other meta metrics (from Line 6 to Line 14). Moreover, meta metrics measuring the dimension of *Errors* will be accumulated for descendants (Line 17), as broken data may not be validated in time.

During the process, an empty meta metric will gather the monitoring metrics of its parents for its children (Line 20). Consider a meta metric (V_i^m), one of its parents without monitoring (V_j^m), and their structural equations (f_i^m and f_j^m). We can substitute f_j^m for unobserved V_j^m in f_i^m, as shown in Eq. (2). Both the parents of V_j^m and those of V_i^m (except V_j^m) show as the parameters of $f_i^{m\prime}$, which is the reason behind Line 20.

$$\begin{aligned} v_i^m &= f_i^m\left(f_j^m\left(\mathbf{pa}_{\mathcal{G}_{skel}}(V_j^m), \mathbf{u}_j^m\right), \mathbf{pa}_{\mathcal{G}_{skel}}(V_i^m) \setminus \{v_j^m\}, \mathbf{u}_i^m\right) \\ &= f_i^{m\prime}\left(\mathbf{pa}_{\mathcal{G}_{skel}}(V_j^m), \mathbf{pa}_{\mathcal{G}_{skel}}(V_i^m) \setminus \{v_j^m\}, \mathbf{u}_i^m, \mathbf{u}_j^m\right) \end{aligned} \tag{2}$$

4.2 Regression-based Hypothesis Testing

The understanding of $P_{\mathbf{m}}$ is restricted by mitigating the failure as soon as possible. Instead of comparing two distributions directly, we reformulate the Intervention Recognition Criterion as hypothesis testing with the following null hypothesis ($\mathbf{H}_0$) for each metric V_i.

$\mathbf{H}_0$ V_i is not an indicator of the root cause, *i.e.*,

$$V_i^{(t)} \sim \mathcal{L}_1\left(V_i^{(t)} \mid \mathbf{pa}^{(t)}(V_i)\right)$$

We utilize the regression technique to calculate the expected distribution $\mathcal{L}_1\left(V_i^{(t)} \mid \mathbf{pa}^{(t)}(V_i)\right)$. A regression model is trained for each variable with data before the fault is detected, performing as a proxy of the corresponding structural equation. Let $\bar{v}_i^{(t)}$ be the regression value for $v_i^{(t)}$. Assuming that the residuals follow an *i.i.d.* normal distribution $N(\mu_{\epsilon,i}, \sigma_{\epsilon,i})$, Eq. (3) measures to what extent a new datum $v_i^{(t)}$ deviates from the expected distribution, denoted as $a_{V_i}^{(t)}$. Eq. (4) further aggregates $a_{V_i}^{(t)}$ for all the available data during the abnormal period as the anomaly score of V_i.

$$a_{V_i}^{(t)} = \left|\frac{\left(v_i^{(t)} - \bar{v}_i^{(t)}\right) - \mu_{\epsilon,i}}{\sigma_{\epsilon,i}}\right| \tag{3}$$

$$s_{V_i} = \max_t a_{V_i}^{(t)} \tag{4}$$

Algorithm 1 Structural Graph Construction

Require: $\mathcal{G}_c$, the call graph; $\mathbf{h} : \mathbf{V}^m \to 2^{\mathbf{V}}$, the mapping from meta metrics $\mathbf{V}^m$ to monitoring metrics

1: $\mathcal{G}_s \leftarrow$ initial the structure graph
2: $\mathcal{G}_{skel} \leftarrow$ construct the skeleton based on $\mathcal{G}_c$
3: **for all** $V_i^m \in \mathbf{V}^m$ in a topological order from $\{V_j^m \mid |\mathbf{Pa}_{\mathcal{G}_{skel}}(V_j^m)| = 0\}$ **do**
4: $\quad \mathbf{Q}_i \leftarrow$ Collect monitoring metrics of $\mathbf{Pa}_{\mathcal{G}_{skel}}(V_j^m)$
5: $\quad \mathbf{C}_i \leftarrow$ Collect monitoring metrics of V_i^m
6: $\quad$ **for** $V_j \in \mathbf{h}(V_i^m)$ **do**
7: $\quad\quad$ **if** V_j is mapped to multiple meta metrics **then**
8: $\quad\quad\quad \mathbf{C}_i \leftarrow \mathbf{C}_i \setminus \{V_j\}$ /* *Prevent self loop of* V_j */
9: $\quad\quad\quad$ **if** V_j is visited for the last time **then**
10: $\quad\quad\quad\quad$ Add edges from the corresponding meta metrics of V_j other than V_i^m to V_j in $\mathcal{G}_s$
11: $\quad\quad\quad\quad \mathbf{Q}_i \leftarrow \mathbf{Q}_i \cup \{V_j\}$ /* *Take it as the proxy of others* */
12: $\quad\quad\quad$ **end if**
13: $\quad\quad$ **end if**
14: $\quad$ **end for** /* *Deal with monitoring metrics that are derived from multiple meta metrics* */
15: $\quad$ Add edges from $\mathbf{Q}_i$ to $\mathbf{C}_i$ in $\mathcal{G}_s$
16: $\quad$ **if** V_i^m represents Errors **then**
17: $\quad\quad$ Update $\mathbf{C}_i$ with monitoring metrics from the Errors-representing meta metrics in $\mathbf{Pa}_{\mathcal{G}_{skel}}(V_j^m)$
18: $\quad$ **end if**/* *Transfer Errors* */
19: $\quad$ **if** $\mathbf{C}_i = \emptyset$ **then**
20: $\quad\quad \mathbf{h}(V_i^m) \leftarrow \mathbf{Q}_i$ /* *Gather monitoring metrics for children* */
21: $\quad$ **else**
22: $\quad\quad \mathbf{h}(V_i^m) \leftarrow \mathbf{C}_i$
23: $\quad$ **end if**
24: **end for**
25: **return** $\mathcal{G}_s$

4.3 Descendant Adjustment

There will be bias in the regression results due to a poor understanding of $\mathcal{L}_1$. We adjust the anomaly score of one metric with those of its descendants. Our intuition is that when both a metric and one of its parents in the CBN is abnormal, we prefer the latter. For example, supplementing extra resources is an actionable mitigation method to restore the low latency. Hence, we assign a higher score for resource utilization (the parents of latency in the CBN) than latency's score.

We summarize the adjustment in Algorithm 2. The children of a metric V_i are first considered (Line 3). We exclude some metrics ($\{V_i \mid s_{V_i} < 3\}$) from the root cause indicators, so called the *three-sigma rule of thumb*. As the failure propagates through them, those metrics will gather anomaly scores from children for the candidate root cause in their ancestors (Line 6). Finally, the anomaly score of V_i (s_{V_i}) will increase by the maximum of descendants' scores just mentioned (Line 12).

Algorithm 2 Descendant Adjustment

Require: $\mathbf{s}$, anomaly scores by Eq. (4)

$S \leftarrow$ a mapping from V_i to the anomaly scores $S(V_i)$ that may be the direct effect of V_i
for $V_i \in \mathbf{V}$ in a topological order from $\{V_j \mid |\mathbf{Ch}(V_j)| = 0\}$ **do**
  $S(V_i) \leftarrow \{s_{V_j} \mid V_j \in \mathbf{Ch}(V_i)\}$
  for $V_j \in \mathbf{Ch}(V_i)$ **do**
    if $s_{V_j} < 3$ **then**
      $S(V_i) \leftarrow S(V_i) \cup S(V_j)$
    end if
  end for
end for/* *Collect direct effects* */
for $V_i \in \mathbf{V}$ **do**
  if $s_{V_j} \geq 3$ **then**
    $s'_{V_i} \leftarrow s_{V_i} + \max(S(V_i))$ /* *Adjust based on descendants* */
  end if
end for
return $\mathbf{s}'$, the adjusted anomaly scores

5 EXPERIMENTS

In this section, we compare the performance of different methods. We first conduct a simulation study to verify their theoretical reliability. The effectiveness is further evaluated on a real-world dataset. All the execution duration is measured on a server with an Intel Xeon E5-2620 CPU @ 2.40GHz (22 cores) and 57GB RAM. We release our code at https://github.com/NetManAIOps/CIRCA. More experiment details are in Appendix C.

5.1 Experimental Setup

5.1.1 Hyperparameters. The shortest sampling interval in our real-world dataset is one minute. Due to the performance consideration, finer monitoring resolution for each metric is uncommon in OSS. Thus, different time series will be pre-processed for the same set of timestamps with the same interval of one minute.

For each fault, let t_d be the **t**ime a fault is **d**etected. We assume that RCA is invoked at $t_d + t_{delay}$ to collect necessary information, while it takes data in the period $[t_d - t_{ref}, t_d + t_{delay}]$ for reference. The data in $(t_d + t_{delay} - t_{test}, t_d + t_{delay}]$ are treated as from $P_{\mathbf{m}}$ while $[t_d - t_{ref}, t_d - t_{test}]$ are taken as fault-free. By default, we use $t_{delay} = 5$ min, $t_{ref} = 120$ min, and $t_{test} = 10$ min in the experiments. The effects of different t_{delay} and t_{ref} will be explored in Section 5.4.1 with the real-world dataset. In the rest of this section, we name a fault with its corresponding data in $[t_d - t_{ref}, t_d + t_{delay}]$ as a *case*.

5.1.2 Evaluation Metrics. Following existing works [17, 25, 29], we evaluate the performance of a method through the recall with the top-k results, denoted as $AC@k$. Eq. (5) shows the definition of $AC@k$, where $\mathcal{F}$ is a set of faults and $R_i(\mathbf{M})$ is the i-th result recommended by the method for each fault $\mathbf{M}$. Eq. (5) is slightly different from the evaluation metrics in the previous works [17, 25, 29], ensuring that $AC@k$ is monotonically non-decreasing with k. 73% of developers only consider the top-5 results of a fault localization technique, according to the survey in [13]. As a result, we present $AC@k$ for $k \leq K = 5$. Moreover, we show the overall performance by $Avg@K = \frac{1}{K}\sum_{k=1}^{K} AC@k$. In terms of efficiency, we record analysis duration per fault, denoted as T in the unit of seconds.

$$AC@k = \frac{1}{|\mathcal{F}|}\sum_{\mathbf{M}\in\mathcal{F}} \frac{|\mathbf{M} \cap \{R_i(\mathbf{M}) \mid i = 1, 2, \cdots, k\}|}{|\mathbf{M}|} \tag{5}$$

5.1.3 Baselines. Each baseline is separated into two steps, namely *graph construction* and *scoring*. Monitoring metrics will be ranked based on the scores calculated in the final step. We classify the scoring step in the recent RCA literature for OSS into three groups: DFS-based, random walk-based, and invariant network-based. In each group, we choose the representative works. Moreover, we choose the graph construction methods adopted in those works as the baseline ones for the first step.

In the graph construction step, the PC algorithm [12] is widely used [4, 14, 16, 25]. We choose Fisher's z-transformation of the partial correlation and G^2 test as the conditional independence tests for PC, denoted as *PC-gauss* and *PC-gsq*, respectively. PCMCI [22] adapts PC for time series, based on which *PCTS* [17] transfers the lagged graph into the one among monitoring metrics. Moreover, the structural graph proposed in this work is denoted as *Structural*.

As for the scoring step, *DFS* traverses the abnormal nodes in the graph, ranking the roots of the sub-graph via anomaly scores [4]. Its variant *DFS-MS* further ranks candidate metrics according to correlation with the SLI [14]. Another variant *DFS-MH* traverses the abnormal sub-graph until a node is not correlated with its parents [15]. The DFS-based methods take the result of anomaly detection as input. We choose z-score used in [14] and SPOT [23] used in [17] as options. These anomaly detection methods are also taken as baselines[2], denoted as *NSigma* and *SPOT*, respectively. Another line of works is random walk-based methods. *RW-Par* calculates the transition probability via partial correlation [17], while *RW-2* is short for the second-order random walk with Pearson correlation [25]. *ENMF*[3] constructs an invariant network based on the ARX model, explicitly modeling the fault propagation [5]. *CRD* further extends *ENMF* with broken cluster identification [18].

5.2 Simulation Study

Three datasets are generated with 50 / 100 / 500 nodes and 100 / 500 / 5,000 edges, respectively, denoted as $\mathcal{D}^N_{Sim}$ where N is the number of nodes. For each dataset, we generate 10 graphs and 100 cases per graph. Evaluation metrics averaged among the 10 graphs will be presented. The parameters of baseline methods are selected to achieve the best AC@5 on the first graph in $\mathcal{D}^{50}_{Sim}$.

5.2.1 Data Generation. We generate time series based on the Vector Auto-regression model, as shown in Eq. (6). $\mathbf{x}^{(t)}$ is a column vector of the metrics at time t. $\mathbf{A}$ is the weighted adjacent matrix encoding the CBN. $A_{ij} \neq 0$ means the j-th metric is a cause of the i-th one, where A_{ij} represents the causal effect, *e.g.*, the memory usage per request. The CBN is enforced to be a connected DAG with only the first node (SLI) having no children. The item $\beta\mathbf{x}^{(t-1)}$ reflects the auto-regression nature of the time series. The final item

[2] Anomaly detection and invariant network-based methods will utilize an empty graph with all the available monitoring metrics but no edges.
[3] We take "ENMF" from their code to prevent abbreviation duplication between Ranking Causal Anomalies [5] and Root Cause Analysis.

$\epsilon^{(t)}$ is Gaussian noises, representing the natural fluctuation due to unobserved variables.

$$\mathbf{x}^{(t)} = \mathbf{A}\mathbf{x}^{(t)} + \beta\mathbf{x}^{(t-1)} + \epsilon^{(t)} \tag{6}$$

To inject a fault $\mathbf{M}$ at time t, we first generate the number of root cause metrics $|\mathbf{M}|$. $|\mathbf{M}| - 1$ follows a Poisson distribution, as it is rare for a fault to affect many metrics directly. For each $V_i \in \mathbf{M}$, the noise item will be altered as $u_i^{(t)} = \epsilon_i^{(t)} + a_i\sigma_i$ for 2 timestamps. The random parameter a_i will make the SLI metric abnormal according to the three-sigma rule of thumb.

5.2.2 Performance Evaluation. Table 1 summarizes the performance of different methods in three simulation datasets. The scoring step of each method uses the graph deduced by $\mathbf{A}$ directly, *i.e.*, $X_j \in \mathbf{Pa}(X_i) \Leftrightarrow \mathbf{A}_{ij} \neq 0$. We choose the linear regression for RHT. Moreover, RHT could achieve the best performance in theory if it regards the parents as $\mathbf{Pa}(X_i^{(t)}) = \mathbf{Pa}^{(t)}(X_i) \cup \left\{X_i^{(t-1)}\right\}$. Such implementation is denoted as *RHT-PG*, where *PG* represents the *perfect graph*. As the linear relation with the perfect graph performs as the best proxy of $\mathcal{L}_1$, we do not consider the descendant adjustment in the simulation study.

RHT-PG approaches the ideal performance, outperforming baseline methods ($p < 0.001$ in t-test for AC@k), which shows the theoretical reliability of our method. There is a gap between the performance of RHT and RHT-PG, which enlarges as the number of nodes increases. This phenomenon illustrates the restriction of Corollary 3.2 that a broken CBN cannot guarantee a correct answer to RCA. On the other hand, RHT-PG is not perfect yet, which may be the result of statistical errors introduced in hypothesis testing with limited faulty data.

5.2.3 Robustness Evaluation. Faults with the same strength may have different effects on the SLI. Yang et al. name such a phenomenon as the *dependency intensity* in cloud systems, *i.e.*, "how much the status of the callee service influences the caller service" [26]. In this simulation study, we further classify faults into three types based on their dependency intensities with the SLI. We evaluate the performance of RCA methods against faults of each type separately.

Eq. (6) can be transformed into $\mathbf{x}^{(t)} = \mathbf{W}(\beta\mathbf{x}^{(t-1)} + \epsilon^{(t)})$, where $\mathbf{W} = (I - \mathbf{A})^{-1}$. Notice that $\mathbf{W}$ is well-defined as $\mathbf{A}$ is generated to be a DAG, which does not have full rank. The element of $\mathbf{W}$ means that x_i will increase by $\mathbf{W}_{ij}$ when x_j increases by 1. Denote the standard deviation of X_i based on data before fault as $\hat{\sigma}_i$. We classify each fault $\mathbf{M}$ in the simulated datasets into three types:

- **Weak** The root cause metrics deviate from the normal status dramatically to make a slight fluctuation in the SLI (the first node), *i.e.*, $(\forall X_i \in \mathbf{M})\ \mathbf{W}_{1i}\hat{\sigma}_i/\hat{\sigma}_1 < 1$;
- **Strong** A slight fluctuation in the root cause metrics can change the SLI dramatically, *i.e.*, $(\forall X_i \in \mathbf{M})\ \mathbf{W}_{1i}\hat{\sigma}_i/\hat{\sigma}_1 > 1$;
- **Mixed** A fault contains metrics with both the above two types or X_i with $\mathbf{W}_{1i}\hat{\sigma}_i/\hat{\sigma}_1 = 1$.

Table 2 shows the results on $\mathcal{D}_{Sim}^{50}$. The results on $\mathcal{D}_{Sim}^{100}$ and $\mathcal{D}_{Sim}^{500}$ are omitted since there are only 4 and 5 strong faults in these two datasets, respectively. RHT and RHT-PG achieve the best results no matter the type of faults, implying that RHT is more robust than baseline methods. Anomaly detection methods have competitive performance with weak faults. Their performance drops in strong faults because root cause metrics may be less abnormal than others. DFS-based methods are sensitive to the results of anomaly detection. Their performance shares a similar trend with anomaly detection methods, from weak faults to strong ones.

5.3 Empirical Study on Oracle Database Data

We further evaluate different methods in a real-world dataset, denoted as $\mathcal{D}_O$. There are 99 cases in $\mathcal{D}_O$. Each case comes from Oracle databases with high AAS faults in a large banking system. We choose the parameters of baseline methods for better AC@5.

5.3.1 Implementation. We manually extract the call graph in an Oracle database instance from the official documentation[5]. After that, we map 197 monitoring metrics to meta metrics in the skeleton. The final structural graph contains 2,641 edges. Oracle database instances may have different sets of metrics. Therefore, we construct the structural graph for each instance with monitored metrics.

In this empirical study, the ground truth graph is unavailable. Hence, we compare graph construction methods for each scoring method, choosing the graph with the highest AC@5. Meanwhile, there is no perfect proxy of $\mathcal{L}_1$ (like the CBN and linear relation in the simulation study). As a result, we fail to include the ideal implementation of RHT (RHT-PG) in the experiment. We choose the Support Vector Regression (SVR) as the regression method for RHT, which will be discussed in Appendix C.4. To alleviate the bias in hypothesis testing, we equip RHT with the descendant adjustment, denoted as *CIRCA*.

5.3.2 Performance Evaluation. CIRCA achieves the best results compared with baseline methods, as shown in Table 3. Random walk-based methods achieve their best performance with PCTS while taking much time to construct the graph. With the structural graph, DFS-based methods and CIRCA recommend root cause metrics within seconds.

We remove components from CIRCA progressively to show their contribution, summarized in Table 4. The result illustrates that both regression-based hypothesis testing and descendant adjustment have a positive effect. Figure 3 compares the proposed structural graph with other graph construction baselines. We exclude anomaly detection and invariant network-based methods from this figure, as they cannot utilize the CBN. Each box in Figure 3 presents the distribution of AC@5 for a scoring method with different parameters. One data point is the best AC@5 from different graph construction parameters with the same scoring ones. The 3 horizontal lines of each box show 25th, 50th, and 75th percentile, while two whiskers extend to minimum and maximum. The proposed structural graph improves AC@5 for DFS-based methods and CIRCA, while PCTS fits random walk-based methods better.

5.3.3 Case Study. Figure 4 presents a failure, where "log file sync" (LFS) is the root cause metric labeled by the database administrators (DBAs). A poor understanding of $\mathcal{L}_1$ puzzles RCA methods. On the one hand, DFS fails to stop at LFS and continues to check "execution

[4] RW-2 is degraded to the first-order random walk with its best parameter, hence having the identical performance to RW-Par.

[5] Oracle Database Concepts. https://docs.oracle.com/cd/E11882_01/server.112/e40540/

Table 1: Performance of different methods in the simulation study. We put the standard deviation in the parentheses behind each evaluation metric. *RHT-PG* represents *RHT* with the *perfect graph*.

Scoring Method	$\mathcal{D}_{Sim}^{50}$ AC@1	$\mathcal{D}_{Sim}^{50}$ AC@5	$\mathcal{D}_{Sim}^{50}$ T (s)	$\mathcal{D}_{Sim}^{100}$ AC@1	$\mathcal{D}_{Sim}^{100}$ AC@5	$\mathcal{D}_{Sim}^{100}$ T (s)	$\mathcal{D}_{Sim}^{500}$ AC@1	$\mathcal{D}_{Sim}^{500}$ AC@5	$\mathcal{D}_{Sim}^{500}$ T (s)
NSigma	0.432(0.05)	0.733(0.03)	0.306(0.00)	0.384(0.05)	0.613(0.03)	0.575(0.01)	0.376(0.04)	0.579(0.03)	2.759(0.03)
SPOT	0.508(0.04)	0.761(0.03)	6.601(0.21)	0.451(0.04)	0.670(0.03)	17.365(1.14)	0.225(0.07)	0.509(0.07)	83.465(10.85)
DFS	0.541(0.04)	0.682(0.05)	0.308(0.00)	0.555(0.03)	0.653(0.03)	0.579(0.01)	0.540(0.03)	0.611(0.02)	2.790(0.06)
DFS-MS	0.515(0.03)	0.682(0.05)	0.502(0.00)	0.517(0.03)	0.652(0.03)	0.964(0.01)	0.191(0.09)	0.542(0.06)	4.665(0.06)
DFS-MH	0.178(0.08)	0.217(0.09)	0.501(0.00)	0.272(0.05)	0.365(0.05)	0.969(0.01)	0.489(0.04)	0.605(0.03)	4.735(0.05)
RW-Par	0.188(0.06)	0.433(0.07)	0.714(0.00)	0.136(0.05)	0.295(0.07)	1.761(0.01)	0.004(0.01)	0.017(0.02)	20.246(0.10)
RW-2^4	0.188(0.06)	0.433(0.07)	0.437(0.00)	0.136(0.05)	0.295(0.07)	1.059(0.01)	0.004(0.01)	0.017(0.02)	10.141(0.11)
ENMF	0.116(0.03)	0.278(0.04)	0.624(0.01)	0.200(0.03)	0.336(0.05)	1.865(0.03)	0.217(0.04)	0.354(0.07)	34.082(0.55)
CRD	0.074(0.02)	0.223(0.04)	4.844(0.03)	0.013(0.01)	0.064(0.02)	6.767(0.10)	0.003(0.01)	0.011(0.01)	46.933(0.74)
RHT	0.598(0.03)	0.880(0.02)	0.338(0.01)	0.535(0.06)	0.749(0.06)	0.658(0.01)	0.510(0.04)	0.644(0.04)	3.326(0.06)
RHT-PG	**0.615**(0.02)	**0.952**(0.01)	0.346(0.00)	**0.631**(0.02)	**0.930**(0.01)	0.665(0.01)	**0.623**(0.03)	**0.823**(0.03)	3.310(0.07)
Ideal	0.617(0.02)	0.999(0.00)		0.633(0.02)	0.999(0.00)		0.634(0.04)	1.000(0.00)	

Table 2: Robustness evaluation on $\mathcal{D}_{Sim}^{50}$. Faults are classified into three types based on their indicators' influence on SLI.

Scoring Method	Weak (n=916) AC@1	Weak (n=916) AC@5	Mixed (n=64) AC@1	Mixed (n=64) AC@5	Strong (n=20) AC@1	Strong (n=20) AC@5
NSigma	0.454	0.753	0.249	0.498	0.000	0.550
SPOT	0.534	0.783	0.293	0.503	0.000	0.550
DFS	0.558	0.707	0.282	0.368	0.550	0.550
DFS-MS	0.531	0.707	0.277	0.368	0.550	0.550
DFS-MH	0.184	0.223	0.069	0.123	0.250	0.250
RW-Par	0.194	0.445	0.142	0.300	0.050	0.300
RW-2^4	0.194	0.445	0.142	0.300	0.050	0.300
ENMF	0.111	0.269	0.124	0.321	0.300	0.550
CRD	0.071	0.207	0.088	0.353	0.150	0.550
RHT	0.613	0.888	0.325	0.730	0.800	1.000
RHT-PG	**0.624**	**0.954**	**0.358**	**0.914**	**1.000**	**1.000**
Ideal	0.627	1.000	0.358	0.995	1.000	1.000

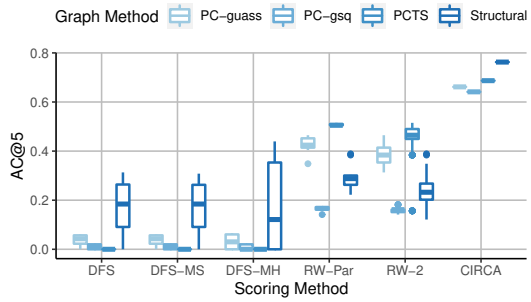


Figure 3: AC@5 for different combinations of ranking methods and graph construction ones

per second" (EPS), missing the desired answer. No baseline method recommends LFS in the top-5 results, except NSigma, ENMF, and

Table 3: Performance of different methods on $\mathcal{D}_O$

Scoring Method	Graph Method	AC@1	AC@5	Avg@5	T (s)
NSigma	Empty	0.323	0.662	0.525	0.472
SPOT	Empty	0.152	0.419	0.296	5.027
DFS	Structural	0.187	0.313	0.271	0.483
DFS-MS	Structural	0.207	0.308	0.275	0.839
DFS-MH	Structural	0.268	0.439	0.372	0.844
RW-Par	PCTS	0.086	0.449	0.290	24.695
RW-2^4	PCTS	0.086	0.449	0.290	24.559
ENMF	Empty	0.111	0.374	0.254	0.771
CRD	Empty	0.035	0.313	0.165	4.787
CIRCA	Structural	**0.404**	**0.763**	**0.603**	0.578
Ideal		0.929	1.000	0.986	

Table 4: Contribution of CIRCA's components on $\mathcal{D}_O$ with the structural graph.

Scoring Method	AC@1	AC@3	AC@5	Avg@5	T (s)
NSigma	0.323	0.586	0.662	0.525	0.472
RHT	0.328	0.601	0.677	0.546	0.576
CIRCA	**0.404**	**0.616**	**0.763**	**0.603**	0.578

CRD. On the other hand, CIRCA assigns a high anomaly score for AAS after revising it from 532.4 (given by NSigma) to 480.2 with regression.

DFS-based methods will drop descendants once meeting an abnormal metric. In contrast, CIRCA scores each metric separately, preventing missing answers like DFS-based methods. Moreover, CIRCA adjusts the anomaly score of LFS with that of the average time of "log file parallel write" (LFPW), *i.e.*, $s'_{LFS} = 7028.6$. This technique helps CIRCA rank LFS ahead of the other metrics.

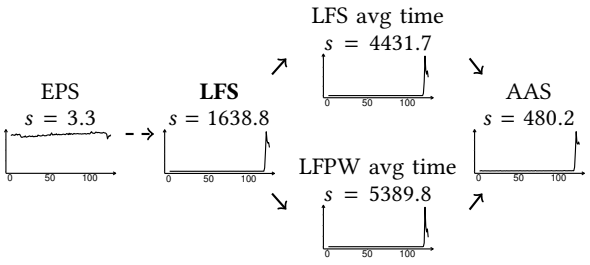


Figure 4: Part of an Oracle database failure, where LFS is the root cause metric labeled by the DBAs. Below each metric name is the score calculated by Eq. (4) and the time series at the same period. Time (horizontal axis) is shown in minutes.

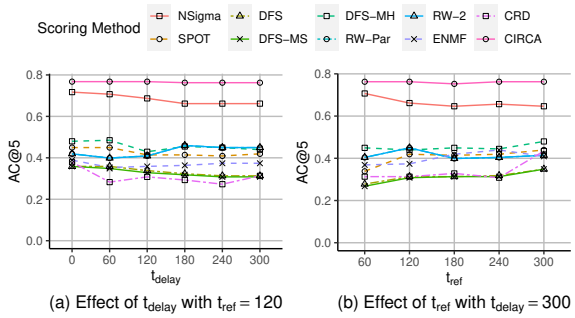


Figure 5: Performance with various hyperparameters on $\mathcal{D}_O$

5.3.4 Lessons Learned. CIRCA outperforms baseline methods on $\mathcal{D}_O$, consistent with the simulation study. Table 4 and Figure 3 further illustrate that each of the 3 proposed techniques has a positive effect.

Though RCA is a difficult task related to Layer 2 of the causal ladder (Corollary 3.2), the knowledge of Layer 1 ($\mathcal{L}_1$) is incomplete. We illustrate the negative effect through a case study. We believe that further advancement in the future has to handle this obstacle explicitly. At present, we prefer CIRCA to pure RHT if deployed. Meanwhile, the effectiveness of the descendant adjustment has to be verified on more real-world datasets.

5.4 Discussion

5.4.1 Hyperparameter Sensitivity. Figure 5 compares RCA methods with different t_{delay} and t_{ref}. CIRCA has stable performance with these two hyperparameters, outperforming baseline methods.

5.4.2 Performance of Existing Methods. RW-Par and RW-2 represent the scoring methods of MicroCause [17] and CloudRanger [25], respectively. However, RW-Par (RW-2) fails to achieve the performance in the corresponding paper. MicroCause utilizes metric priority provided by operators, which is unavailable for RW-Par. On the other hand, CloudRanger achieves its best result with a sampling interval of 5 seconds. The coarse monitoring frequency of $\mathcal{D}_O$ may explain the poor performance of RW-2.

As stated by Corollary 3.2, knowledge of Layer 2 (such as **Pa**) is necessary for RCA. The invariant network-based methods utilize the observation data only (Layer 1). Their unsatisfying performance illustrates the restriction of CHT [1].

5.4.3 Feasibility. **Graph Construction.** The construction of the proposed structural graph requires system architecture and a mapping from monitoring metrics to the targets to be monitored. The former is usually in the form of documentation. We argue that a metric is neither insightful nor actionable unless operators understand its underlying meaning. Operators need to classify each distinct metrics only once to obtain the mapping. The mapping can be shared among similar instances of the same type (like Oracle database instances).

Scalability. As shown in Table 1, RHT's time cost grows around linearly with the size of the dataset. Moreover, the design of CIRCA supports horizontal scalability to handle large-scale systems via adding computing resources, as each metric is scored separately. We plan to train the regression models offline to speed up online analysis. Mature parallel programming frameworks, such as Apache Spark, may further help accelerate CIRCA.

6 RELATED WORKS

Root Cause Analysis. Corollary 3.2 explains that graph construction is a common step in the RCA literature for online service system operation. DFS-based methods [4, 14, 15] traverse abnormal subgraph, which is sensitive to anomaly detection results. Some works adopt random walk [16, 17, 25] or PageRank [24] to score candidate root cause indicators, lacking explainability. Another line of works is invariant network-based methods [5, 18]. As these works adopt the pair-wise manner to learn the invariant relations, it is hard for them to reach the knowledge of RCA, restricted by CHT [1]. No methods above utilize causal inference. Sage [8] conducts counterfactual analysis to locate root causes without a formal formulation. Corollary 3.3 states that counterfactual analysis is unnecessary. Hence, we did not include this method as a baseline. Meanwhile, CHT [1] indicates that it can be hard to conduct counterfactual analysis even with a CBN.

The definition of root cause analysis varies with the scenario in the literature. Some applications require an answer beyond the data, taking RCA as a classification task with supervised learning [28]. For homogeneous devices or services, operators are interested in the common features [30]. Accordingly, a multi-dimensional root cause analysis is conducted. In contrast, we treat the observed projection of a fault as the desired answer. The Intervention Recognition Criterion further relates RCA in this work with *contextual anomaly detection* [3], treating parents in the CBN as the context for each variable. We take complex contextual anomaly detection methods as future work.

Causal Discovery. The task to obtain the CBN is named *causal discovery*. We refer the readers to a recent survey [10] for a thorough discussion. NOTEARS [32] converts the DAG search problem from the discrete space into a continuous one. Following NOTEARS, some recent works are based on gradient descent [11].

Although causal discovery has its sound theory, the CBN discovered from data directly is not explainable for human operators.

In contrast, some works obtain the CBN based on domain knowledge. MicroHECL [15] traces the fault along with traffic, latency, or error rate in the call graph. Meanwhile, Sage [8] constructs the CBN among latency and machine metrics. The structural graph proposed in this work is compatible with the assumptions in these two works, extending the kinds of meta metrics.

7 CONCLUSION AND FUTURE WORK

Root cause analysis (RCA) is an essential task for OSS operations. In this work, we formulate RCA as a new causal inference task named *intervention recognition*, based on which, we further obtain the Intervention Recognition Criterion to find the root cause. We believe such a formulation bridge two well-studied fields (RCA and causal inference) and provide a promising new direction for the critical-yet-hard-to-solve RCA problem in OSS.

To apply such a criterion in OSS, we propose a novel causal inference-based RCA method, *CIRCA*. CIRCA consists of three techniques, namely structural graph construction, regression-based hypothesis testing, and descendant adjustment. We verify the theoretical reliability of CIRCA in the simulation study. Moreover, CIRCA also outperforms baseline methods in a real-world dataset.

In the future, we plan to include faulty data for regression. We hope that diverse data can help overcome the limited understanding of the system's normal status. This work rests on a set of assumptions that a real application may not satisfy. For example, some meta metrics do not have corresponding monitoring metrics in the skeleton we construct for the Oracle database. As a result, they can imply common exogenous parents of the downstream monitoring metrics, violating the Markovian assumption. Explicitly modeling these hidden meta metrics may improve RCA performance. Meanwhile, the retrospect of analysis mistakes may also point to the lack of monitoring. Beyond the analysis framework in this work, discoveries on the underlying mechanism of OSS can also help climb the ladder of causation for the RCA task.

ACKNOWLEDGMENTS

We thank Li Cao, Zhihan Li, and Yuan Meng for their helpful discussions on this work, thank Xianglin Lu for her data preparation work, and thank Xiangyang Chen, Duogang Wu, and Xin Yang for sharing their knowledge on Oracle databases. This work is supported by the National Key R&D Program of China under Grant 2019YFB1802504, and the State Key Program of National Natural Science of China under Grant 62072264.

REFERENCES

[1] Elias Bareinboim, Juan D. Correa, Duligur Ibeling, and Thomas Icard. 2022. *On Pearl's Hierarchy and the Foundations of Causal Inference* (1 ed.). Association for Computing Machinery, 507–556.
[2] Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. 2016. *Site Reliability Engineering* (first ed.). O'Reilly Media, Inc.
[3] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly Detection: A Survey. *ACM Comput. Surv.* 41, 3, Article 15 (jul 2009), 58 pages.
[4] P. Chen, Y. Qi, P. Zheng, and D. Hou. 2014. CauseInfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems. In *INFOCOM*. 1887–1895.
[5] Wei Cheng, Kai Zhang, Haifeng Chen, Guofei Jiang, Zhengzhang Chen, and Wei Wang. 2016. Ranking Causal Anomalies via Temporal and Dynamical Analysis on Vanishing Correlations. In *KDD*. 805–814.
[6] Amin Dhaou, Antoine Bertoncello, Sébastien Gourvénec, Josselin Garnier, and Erwan Le Pennec. 2021. Causal and Interpretable Rules for Time Series Analysis. In *KDD*. 2764–2772.
[7] Silvery Fu, Saurabh Gupta, Radhika Mittal, and Sylvia Ratnasamy. 2021. On the Use of ML for Blackbox System Performance Prediction. In *NSDI*. 763–784.
[8] Yu Gan, Mingyu Liang, Sundar Dev, David Lo, and Christina Delimitrou. 2021. Sage: Practical and Scalable ML-Driven Performance Debugging in Microservices. In *ASPLOS*. 135–151.
[9] Janos Gertler. 1998. *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker.
[10] Ruocheng Guo, Lu Cheng, Jundong Li, P. Richard Hahn, and Huan Liu. 2020. A Survey of Learning Causality with Data: Problems and Methods. *ACM Comput. Surv.* 53, 4 (jul 2020), 37 pages.
[11] Yue He, Peng Cui, Zheyan Shen, Renzhe Xu, Furui Liu, and Yong Jiang. 2021. DARING: Differentiable Causal Discovery with Residual Independence. In *KDD*. 596–605.
[12] Markus Kalisch, Martin Mächler, Diego Colombo, Marloes H. Maathuis, and Peter Bühlmann. 2012. Causal Inference Using Graphical Models with the R Package pcalg. *Journal of Statistical Software* 47, 11 (2012), 1–26.
[13] Pavneet Singh Kochhar, Xin Xia, David Lo, and Shanping Li. 2016. Practitioners' Expectations on Automated Fault Localization. In *ISSTA*. 165–176.
[14] Jinjin Lin, Pengfei Chen, and Zibin Zheng. 2018. "Microscope: Pinpoint Performance Issues with Causal Graphs in Micro-service Environments". In *Service-Oriented Computing*. 3–20.
[15] Dewei Liu, Chuan He, Xin Peng, Fan Lin, Chenxi Zhang, Shengfang Gong, Ziang Li, Jiayu Ou, and Zheshun Wu. 2021. MicroHECL: High-Efficient Root Cause Localization in Large-Scale Microservice Systems. In *ICSE-SEIP*. 338–347.
[16] Meng Ma, Jingmin Xu, Yuan Wang, Pengfei Chen, Zonghua Zhang, and Ping Wang. 2020. AutoMAP: Diagnose Your Microservice-Based Web Applications Automatically. In *WWW*. 246–258.
[17] Yuan Meng, Shenglin Zhang, Yongqian Sun, Ruru Zhang, Zhilong Hu, Yiyin Zhang, Chenyang Jia, Zhaogang Wang, and Dan Pei. 2020. Localizing Failure Root Causes in a Microservice through Causality Inference. In *IWQoS*. 1–10.
[18] Jingchao Ni, Wei Cheng, Kai Zhang, Dongjin Song, Tan Yan, Haifeng Chen, and Xiang Zhang. 2017. Ranking Causal Anomalies by Modeling Local Propagations on Networked Systems. In *ICDM*. 1003–1008.
[19] Paolo Notaro, Jorge Cardoso, and Michael Gerndt. 2021. A Survey of AIOps Methods for Failure Management. *ACM Trans. Intell. Syst. Technol.* 12, 6, Article 81 (nov 2021), 45 pages.
[20] Judea Pearl. 2009. *Causality : models, reasoning, and inference* (second ed.). Cambridge University Press.
[21] Jamie Pool, Ebrahim Beyrami, Vishak Gopal, Ashkan Aazami, Jayant Gupchup, Jeff Rowland, Binlong Li, Pritesh Kanani, Ross Cutler, and Johannes Gehrke. 2020. Lumos: A Library for Diagnosing Metric Regressions in Web-Scale Applications. In *KDD*. 2562–2570.
[22] Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. 2019. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances* 5, 11 (2019), eaau4996.
[23] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. 2017. Anomaly Detection in Streams with Extreme Value Theory. In *KDD*. 1067–1075.
[24] Hanzhang Wang, Zhengkai Wu, Huai Jiang, Yichao Huang, Jiamu Wang, Selcuk Kopru, and Tao Xie. 2021. Groot: An Event-graph-based Approach for Root Cause Analysis in Industrial Settings. In *ASE*. 419–429.
[25] Ping Wang, Jingmin Xu, Meng Ma, Weilan Lin, Disheng Pan, Yuan Wang, and Pengfei Chen. 2018. CloudRanger: Root Cause Identification for Cloud Native Systems. In *CCGRID*. 492–502.
[26] Tianyi Yang, Jiacheng Shen, Yuxin Su, Xiao Ling, Yongqiang Yang, and Michael R. Lyu. 2021. AID: Efficient Prediction of Aggregated Intensity of Dependency in Large-scale Cloud Systems. In *ASE*. 653–665.
[27] Liuyi Yao, Zhixuan Chu, Sheng Li, Yaliang Li, Jing Gao, and Aidong Zhang. 2021. A Survey on Causal Inference. *ACM Trans. Knowl. Discov. Data* 15, 5, Article 74 (may 2021), 46 pages.
[28] Jaehyuk Yi and Jinkyoo Park. 2021. Semi-Supervised Bearing Fault Diagnosis with Adversarially-Trained Phase-Consistent Network. In *KDD*. 3875–3885.
[29] Guangba Yu, Pengfei Chen, Hongyang Chen, Zijie Guan, Zicheng Huang, Linxiao Jing, Tianjun Weng, Xinmeng Sun, and Xiaoyun Li. 2021. MicroRank: End-to-End Latency Issue Localization with Extended Spectrum Analysis in Microservice Environments. In *WWW*. 3087–3098.
[30] Xu Zhang, Chao Du, Yifan Li, Yong Xu, Hongyu Zhang, Si Qin, Ze Li, Qingwei Lin, Yingnong Dang, Andrew Zhou, Saravanakumar Rajmohan, and Dongmei Zhang. 2021. HALO: Hierarchy-Aware Fault Localization for Cloud Systems. In *KDD*. 3948–3958.
[31] Nengwen Zhao, Junjie Chen, Xiao Peng, Honglin Wang, Xinya Wu, Yuanzong Zhang, Zikai Chen, Xiangzhong Zheng, Xiaohui Nie, Gang Wang, Yong Wu, Fang Zhou, Wenchi Zhang, Kaixin Sui, and Dan Pei. 2020. Understanding and Handling Alert Storm for Online Service Systems. In *ICSE-SEIP*. 162–171.
[32] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. 2018. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In *NIPS*, Vol. 31. 9472–9483.

A PROOF OF THEOREM 3.1

We define *identifiable intervention recognition* (IIR) as follows:

Definition A.1 (Identifiable Intervention Recognition, IIR). Identifiable intervention recognition is to find out a set of potential interventions $\{\mathbf{m}' \mid \mathcal{L}_1(\mathbf{V} \mid do(\mathbf{m}')) \equiv P_{\mathbf{m}}\}$ (*i.e., identifiable interventions*).

With Definition A.1, we have Lemma A.2 and Lemma A.3.

Lemma A.2. *The knowledge of IIR can be derived from $\mathcal{L}_2$.*

Proof of Lemma A.2. Denote C as the equivalence classes defined by $\mathcal{L}_2$ among $\mathcal{F} = \bigcup_{\mathbf{M}\in 2^{\mathbf{V}}} Val(\mathbf{M})$, where $\mathcal{F}$ is all possible interventions, including no intervention. For each equivalent class $c \in C$, c is a set of interventions $[\mathbf{m}]$ which leads to the same distribution over $\mathbf{V}$:

$$[\mathbf{m}] = \{\mathbf{m}' \mid \mathcal{L}_1(\mathbf{V} \mid do(\mathbf{m}')) \equiv P_{\mathbf{m}}\}$$

Denote the distribution of V under $[\mathbf{m}]$ as $\mathcal{L}_2^{-1}(P_{\mathbf{m}}) = [\mathbf{m}]$. Denote $\mathcal{L}_2'([\mathbf{m}]) = P_{\mathbf{m}}$, where $[\mathbf{m}] \in C$. For any $\mathbf{m}_1, \mathbf{m}_2 \in \mathcal{F}$, we always have:

$$P_{\mathbf{m}_1} \equiv P_{\mathbf{m}_2} \rightarrow [\mathbf{m}_1] = [\mathbf{m}_2]$$

Hence, $\mathcal{L}_2'$ is a one-to-one correspondence and have its inverse mapping, denoted as $\mathcal{L}_2^{-1}(P_{\mathbf{m}}) = [\mathbf{m}]$.

When an intervention occurs, based on $P_{\mathbf{m}} \in \mathcal{L}_2(\mathcal{F})$, $\mathcal{L}_2^{-1}(P_{\mathbf{m}})$ is the set of targets of IIR. Hence, the knowledge of IIR can be derived from $\mathcal{L}_2$. □

Lemma A.3. *The knowledge of IIR encodes $\mathcal{L}_2$.*

Proof of Lemma A.3. For any valid $P_{\mathbf{m}} \in \mathcal{L}_2(\mathcal{F})$, IIR will produce a set of possible interventions $\{\mathbf{m}' \mid \mathcal{L}_2(\mathbf{m}') \equiv P_{\mathbf{m}}\}$. Hence, the knowledge of IIR can be extended to the mapping $\mathcal{L}_2^{-1}$ that maps $P_{\mathbf{m}}$ to the element of C. Meanwhile, $\mathcal{L}_2^{-1}$ is the inverse mapping of $\mathcal{L}_2'$ and $\mathcal{L}_2$ can be derived from $\mathcal{L}_2'$. Hence, the knowledge of IIR encodes $\mathcal{L}_2$. □

Based on Lemma A.2 and Lemma A.3, the knowledge of IIR is equivalent to $\mathcal{L}_2$. Then we have Theorem A.4.

Theorem A.4. *The knowledge of IIR is at the second layer of the causal ladder.*

For an SCM with a CBN, we get Lemma A.5.

Lemma A.5. *For a given SCM $\mathcal{M}$ with a CBN $\mathcal{G}$, let $\mathbf{Pa}(V_i)$ be the parents of V_i in $\mathcal{G}$. $P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m}))$ can be reduced to the form defined in Eq.* (7).

$$P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m})) = \begin{cases} P(V_i \mid \mathbf{pa}(V_i), do(v_i)), & V_i \in \mathbf{M} \\ P(V_i \mid \mathbf{pa}(V_i)), & V_i \notin \mathbf{M} \end{cases} \quad (7)$$

Proof of Lemma A.5. Given a variable V_i, the intervened variables $\mathbf{M}$ can be separated into three parts: 1) $\mathbf{M}_{Pa} = \mathbf{M} \cap \mathbf{Pa}(V_i)$, 2) $\mathbf{M}_i = \mathbf{M} \cap \{V_i\}$, and 3) $\mathbf{M}_{Other} = \mathbf{M} \setminus \mathbf{Pa}(V_i) \setminus \{V_i\}$. Hence,

- $\mathbf{M} = \mathbf{M}_{Pa} \cup \mathbf{M}_i \cup \mathbf{M}_{Other}$,
- $\mathbf{M}_{Pa} \cap \mathbf{M}_i = \mathbf{M}_{Pa} \cap \mathbf{M}_{Other} = \mathbf{M}_i \cap \mathbf{M}_{Other} = \emptyset$, and
- there is no arrow from $\mathbf{M}_{Other}$ to V_i in $\mathcal{G}$.

Case 1. With $V_i \in \mathbf{M}$, denote the interventional SCM of $\mathcal{M}$ with $do(\mathbf{m})$ as $\mathcal{M}'$. $\mathcal{M}'$ replaces f_i in $\mathcal{M}$ with $v_i \leftarrow m_i$ [20]. As a result, other variables cannot affect V_i any longer. Hence, $do(v_i)$ makes $P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m}))$ and $P(V_i \mid \mathbf{pa}(V_i), do(v_i))$ equivalent by

$$P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m})) = P(V_i \mid do(v_i)) = P(V_i \mid \mathbf{pa}(V_i), do(v_i))$$

Case 2. With $V_i \notin \mathbf{M}$, we get $\mathbf{M}_i = \emptyset$ and $\mathbf{M} = \mathbf{M}_{Pa} \cup \mathbf{M}_{Other}$ for the equation below

$$P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m})) = P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m}_{Pa}), do(\mathbf{m}_{Other}))$$

Since $\mathcal{G}$ is a CBN, the "Parents do/see" condition [1] states that we can replace $\mathbf{pa}(V_i)$ with $do(\mathbf{pa}(V_i))$.

$$P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m})) = P(V_i \mid do(\mathbf{pa}(V_i)), do(\mathbf{m}_{Other})) \quad (8)$$

As we already take $do(\mathbf{m}_{Pa})$ as the condition, the "Missing-link" condition [1] ensures that we can drop $do(\mathbf{m}_{Other})$.

$$P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m})) = P(V_i \mid do(\mathbf{pa}(V_i))) \quad (9)$$

With the "Parents do/see" condition [1] again, we obtain

$$P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m})) = P(V_i \mid \mathbf{pa}(V_i)) \quad (10)$$

Combining the two cases above provides the final conclusion. □

Proof of Theorem 3.1. Given an intervention $\mathbf{m}$ and any identifiable intervention $\mathbf{m}'$ provided by IIR, $P(\mathbf{V} \mid do(\mathbf{m})) \equiv P(\mathbf{V} \mid do(\mathbf{m}'))$. Hence, $P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m})) \equiv P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m}'))$ for any $V_i \in \mathbf{V}$. Notice that the corresponding assignment for an intervention is just encoded in the interventional distribution. As a result, $(\forall x \in \mathbf{m}, x' \in \mathbf{m}')X = X' \rightarrow x = x'$.

Assume that $\mathbf{m}$ is different from $\mathbf{m}'$, *e.g.*, $(\exists X \in \mathbf{V})X \in \mathbf{M} \wedge X \notin \mathbf{M}'$. With Lemma A.5, we have $P(X \mid \mathbf{pa}(X), do(x)) \equiv P(X \mid \mathbf{pa}(X))$, which violates the Faithfulness assumption. It is the same for the case $(\exists X \in \mathbf{V})X \notin \mathbf{M} \wedge X \in \mathbf{M}'$. Hence, IIR can distinguish $\mathbf{m}$ from other interventions, providing the same answer as IR.

According to Theorem A.4, we reach the conclusion that the knowledge of IR is at the second layer of the causal ladder. □

B PROOF OF THEOREM 3.4

Proof of Theorem 3.4. Notice that $P_{\mathbf{m}}(V_i \mid \mathbf{pa}(V_i)) = P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m}))$, while $\mathcal{L}_1(V_i \mid \mathbf{pa}(V_i)) = P(V_i \mid \mathbf{pa}(V_i))$.

Case 1. With $V_i \in \mathbf{M}$, we get $\mathbf{M}_i = \{V_i\}$. Under the Faithfulness assumption, Eq. (11) must hold, while Lemma A.5 provides $P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m})) = P(V_i \mid \mathbf{pa}(V_i), do(v_i))$.

$$P(V_i \mid \mathbf{pa}(V_i), do(v_i)) \neq P(V_i \mid \mathbf{pa}(V_i)) \quad (11)$$

Hence,

$$V_i \in \mathbf{M} \Rightarrow P_{\mathbf{m}}(V_i \mid \mathbf{pa}(V_i)) \neq \mathcal{L}_1(V_i \mid \mathbf{pa}(V_i))$$

Case 2. With $V_i \notin \mathbf{M}$, Lemma A.5 provides $P(V_i \mid \mathbf{pa}(V_i), do(\mathbf{m})) = P(V_i \mid \mathbf{pa}(V_i))$. Hence,

$$V_i \notin \mathbf{M} \Rightarrow P_{\mathbf{m}}(V_i \mid \mathbf{pa}(V_i)) = \mathcal{L}_1(V_i \mid \mathbf{pa}(V_i))$$

Its contrapositive stands as well,

$$P_{\mathbf{m}}(V_i \mid \mathbf{pa}(V_i)) \neq \mathcal{L}_1(V_i \mid \mathbf{pa}(V_i)) \Rightarrow V_i \in \mathbf{M}$$

, which is the converse proposition of Case 1.

In conclusion, $V_i \in \mathbf{M} \Leftrightarrow P_{\mathbf{m}}(V_i \mid \mathbf{pa}(V_i)) \neq \mathcal{L}_1(V_i \mid \mathbf{pa}(V_i))$. □

C IMPLEMENTATION DETAILS

C.1 Baseline Methods

Most of the code in this work is written in Python, while we adopt the R package pcalg [12] for the PC algorithm. We utilize process-based parallel programming to isolate errors only.

NSigma calculates $\max_t \frac{|v_i^{(t)} - \mu_i|}{\sigma_i}$. We adopt the authors' implementation[6] for SPOT [23] while re-implementing ENMF [5] in Python based on the authors' MATLAB implementation[7]. The other baseline methods are not publicly available. We implement them by our understanding.

C.2 Simulation Data Generation

We generate the simulation datasets based on the Vector Auto-regression model, as shown in Eq. (6). Following existing work [11, 32], the value of non-zero elements in the weighted adjacent matrix, A_{ij}, is uniformly sampled from $(-2.0, -0.5) \cup (0.5, 2.0)$. For the second item $\beta \mathbf{x}^{(t-1)}$, we set $\beta = 0.1$ in the experiment. Finally, we sample the standard deviations from an exponential distribution for the zero-mean Gaussian noises $\epsilon^{(t)}$.

The structure of A is generated in two steps, as shown in Algorithm 3. We first generate a tree to ensure that the graph is a connected DAG. Then, the other edges are inserted randomly.

Algorithm 3 Graph Generation in the Simulation Study

Require: N_{node}, the number of nodes; N_{edge}, the number of edges

$\mathcal{G} \leftarrow (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \{1, 2, \cdots, N_{node}\}$
for $i = 2, \cdots, N_{node}$ **do**
  $j \leftarrow$ choose one node from $\{1, 2, \cdots, i-1\}$ randomly
  Add the edge $i \rightarrow j$ into $\mathbf{E}$
end for
for $k = N_{node}, N_{node}+1, \cdots, N_{edge}$ **do**
  $i, j \leftarrow$ sample $i, j \in \mathbf{V}$ randomly, *s.t.*, $i > j \wedge (i \rightarrow j) \notin \mathbf{E}$
  Add the edge $i \rightarrow j$ into $\mathbf{E}$
end for
return $\mathcal{G}$

C.3 Structural Graph Construction

In the empirical study, we construct the structural graph for the Oracle database instances. Figure 6 shows the SQL processing and memory structures in the call graph. Figure 4 further shows part of the graph among metrics. We drop metrics not included in the final structural graph, as our knowledge fails to cover them. Baseline methods only use labeled metrics for a fair comparison.

C.4 Regression Method Selection

Table 5 shows RHT's performance with several regression methods. RHT with the linear regression (*Linear*) has unsatisfying performance, as the relations among real-world variables are seldom linear. Support Vector Regression (*SVR*) with the sigmoid kernel, which is non-linear, improves the performance. Fu et al. provide a way to predict distribution based on Random Forest (RF) and Mixture Density Networks (MDN), respectively, instead of a single value [7]. Hence, RHT combined with RF or MDN can measure the deviation for a new datum against the predicted distribution. However, these two methods perform worse than the simple linear regression due to a limited understanding of the normal status, as shown in Figure 1. As a result, we choose SVR in the empirical study.

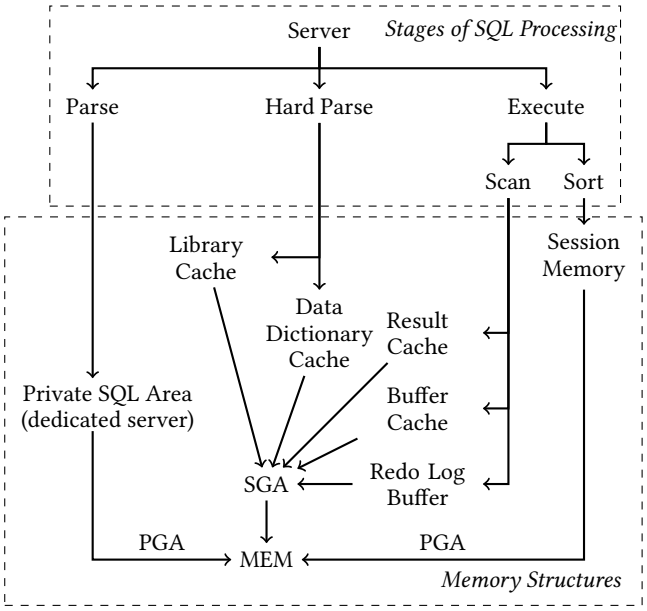


Figure 6: Part of the Oracle database call graph

Table 5: RHT with different regression methods on $\mathcal{D}_O$

Regression Method	AC@1	AC@3	AC@5	Avg@5	T (s)
Linear	0.197	0.424	0.556	0.409	0.559
SVR	**0.328**	**0.601**	**0.677**	**0.546**	0.834
RF	0.202	0.394	0.525	0.382	22.065
MDN	0.111	0.212	0.253	0.195	694.329

[6] https://github.com/Amossys-team/SPOT
[7] https://github.com/chengw07/CausalRanking