Incremental Causal Graph Learning for Online Root Cause Analysis

Dongjie Wang[†]
University of Central Florida
FL, USA
dwang@nec-labs.com

Zhengzhang Chen[§]
NEC Laboratories America Inc.
NJ, USA
zchen@nec-labs.com

Yanjie Fu[§]
University of Central Florida
FL, USA
yanjie.fu@ucf.edu

Yanchi Liu
NEC Laboratories America Inc.
NJ, USA
yanchi@nec-labs.com

Haifeng Chen
NEC Laboratories America Inc.
NJ, USA
haifeng@nec-labs.com

ABSTRACT

The task of root cause analysis (RCA) is to identify the root causes of system faults/failures by analyzing system monitoring data. Efficient RCA can greatly accelerate system failure recovery and mitigate system damages or financial losses. However, previous research has mostly focused on developing offline RCA algorithms, which often require manually initiating the RCA process, a significant amount of time and data to train a robust model, and then being retrained from scratch for a new system fault.

In this paper, we propose CORAL, a novel online RCA framework that can automatically trigger the RCA process and incrementally update the RCA model. CORAL consists of Trigger Point Detection, Incremental Disentangled Causal Graph Learning, and Network Propagation-based Root Cause Localization. The Trigger Point Detection component aims to detect system state transitions automatically and in near-real-time. To achieve this, we develop an online trigger point detection approach based on multivariate singular spectrum analysis and cumulative sum statistics. To efficiently update the RCA model, we propose an incremental disentangled causal graph learning approach to decouple the state-invariant and state-dependent information. After that, CORAL applies a random walk with restarts to the updated causal graph to accurately identify root causes. The online RCA process terminates when the causal graph and the generated root cause list converge. Extensive experiments on three real-world datasets demonstrate the effectiveness and superiority of the proposed framework.

CCS CONCEPTS

• **Computing methodologies** → **Causal reasoning and diagnostics**; **Online learning settings**; *Unsupervised learning*.

†Work done during an internship at NEC Laboratories America.
§Corresponding author.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599392

KEYWORDS

Root Cause Analysis; AIOps; Causal Discovery; Trigger Point Detection; Incremental Learning; Disentangled Graph Learning

ACM Reference Format:
Dongjie Wang[†], Zhengzhang Chen[§], Yanjie Fu[§], Yanchi Liu, and Haifeng Chen. 2023. Incremental Causal Graph Learning for Online Root Cause Analysis. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3580305.3599392

1 INTRODUCTION

Root Cause Analysis (RCA) aims to identify the underlying causes of system faults (a.k.a., anomalies, malfunctions, errors, failures) based on system monitoring data [4, 28]. RCA has been widely used in IT operations, telecommunications, industrial process control, *etc.*, because a fault in these systems can greatly lower user experiences, and cause huge losses. For instance, in 2021, an intermittent outage of Amazon Web Services caused over 210 millions in losses [15].

Previous RCA studies [31, 34, 38, 52] have focused primarily on developing effective **offline** methods for root cause localization. A key component of many data-driven offline RCA algorithms (see Fig. 1), especially the causal discovery based RCA methods [26, 31, 38], is to learn the causal structure or causal graph that profiles the causal relations between system entities and system Key Performance Indicator (KPI) based on historical data, so that the operators can trace back the root causes based on the built causal graph. For instance, Ikram *et al.* [26] utilized historical multivariate monitoring data to construct causal graphs using the conditional interdependence test, and then applied causal intervention to identify the root causes of a microservice system.

However, there are three limitations in the traditional offline causal discovery based RCA workflow (**Fig. 1**). First, for a new system's faults, we need to retrain/rebuild the model from scratch. Second, the causal graph learning component is often time-consuming and requires a large amount of historical data to train a robust model. Third, it often requires the operators to manually initiate the RCA process when they observe a system fault. As a result, it's often too late to mitigate any damage or loss caused by a system fault. These motivate us to ask the following questions: 1) Is it possible to perform a causal discovery-based RCA task efficiently? 2) How can we identify the root cause as early as possible? 3) Can

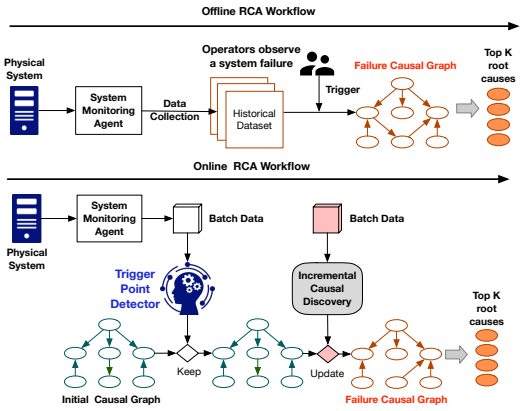


Figure 1: Comparison between a traditional offline RCA workflow and the proposed CORAL online RCA workflow.

we deploy the RCA algorithm online for the streaming data? 4) If deployed online, can we avoid the time-consuming retraining of the RCA model from scratch every time a system fault occurs?

In recent years, a very promising means for learning streaming data has emerged through the concept of incremental learning (*aka* continual learning or lifelong learning) [18]. Such incremental learning models rely on a compact representation of the already observed signals or an implicit data representation due to limited memory resources. In our RCA task, if we can incrementally update the RCA model or causal graph for each batch of the streaming data, it virtually accelerates the RCA process. More importantly, we don't need to wait until the system fault occurs to trigger the RCA process, or we might even be able to trigger an early RCA to mitigate the damages and losses. Thus, there exists a vital need for methods that can incrementally learn the RCA model and automatically trigger the RCA process.

Enlightened by incremental learning, this paper aims to incrementally update the causal graph from streaming system monitoring data for accurately identifying root causes when a system failure or fault occurs. Formally, given the initial causal graph learned from historical data, and the streaming system monitoring data including entity metrics and KPI data, our goal is to automatically initiate the RCA process when a system fault occurs, incrementally update the initial causal graph by considering each batch of data sequentially, and efficiently identify the top K nodes (*i.e.*, system entities) in the updated causal graph that is most relevant to system KPI. There are two major challenges in this task:

- **Challenge 1: Identify the transition points between system states to initiate root cause analysis.** As aforementioned, in traditional RCA, the operators often manually initiate the root cause procedure after a system fault occurs. To mitigate the damages or losses, in an online setting, we need to automatically detect the system state changes caused by the system fault and trigger the RCA process. The challenge is how to identify the transition points of system states early if the fault does not affect the system KPI, but only affects some root cause system entities at the early stage.
- **Challenge 2: Incrementally update the causal graph model in an efficient manner.** After the transition/trigger points are detected, we can not directly apply the old RCA model or causal graph to identify the root causes, since the old causal graph only contains the causal relations learned from the previous system state data. Although some inherent system dependencies will never change over time [36] (*i.e.*, system state-invariant causation), other causal dependencies may be highly dependent on the system state (*i.e.*, system state-dependent causation). The challenge is how to identify the system state-invariant causation from the old model and quickly learn the state-dependent causation from the new batches of data for accelerating causal graph learning.

To address these challenges, in this paper, we propose CORAL, a novel incremental causal graph learning framework, for online root cause localization. CORAL consists of three main steps: 1) Trigger Point Detection; 2) Incremental Disentangled Causal Graph Learning; and 3) Network Propagation-Based Root Cause Localization. In particular, the first step of CORAL is to detect the transition points between system states in real time based on system entity metrics and KPI data. To detect trigger points with less delay, we develop an online trigger point detection algorithm based on multivariate singular spectrum analysis and cumulative sum statistics. These points are then used to trigger incremental causal graph learning. Assuming that as the system state transitions, the underlying causal structure partially changes and evolves over time instead of shifting abruptly and significantly. Based on this assumption, we propose an incremental disentangled causal graph learning model to efficiently learn causal relations by decoupling state-invariant and state-dependent causations. After that, we apply a random walk with restarts to model the network propagation of system faults to accurately identify root causes. The online root cause localization process terminates for the current system fault when the learned causal graph and the generated root cause list converge. To summarize, our main contributions are three-fold:

- ***Problem***: We investigate the novel problem of online root cause localization. Remarkably, we propose to solve this problem by automatic trigger point detection and incremental causal structure learning.
- ***Algorithms***: We propose a principled framework CORAL, which integrates a new family of disentangled representation learning (*i.e.*, causal graph disentanglement), online trigger point detection, and incremental causal discovery.
- ***Evaluations***: We perform extensive experiments on three real-world datasets to validate the effectiveness of our approach. The experimental results demonstrate the superior performance of CORAL over the state-of-the-art methods on root cause localization.

2 PRELIMINARIES

System Key Performance Indicator (KPI) is a monitoring time series that indicates the system status. For example, in a microservice system, latency is a KPI to measure the system status. The lower (higher) a system's latency is, the better (worse) its performance is. **Entity Metrics** are multivariate time series collected by monitoring numerous system entities/components. For example, in a microservice system, a system entity can be a physical machine, container, virtual machine, pod, and so on. The system metrics include CPU utilization, memory consumption, disk IO utilization, etc. System

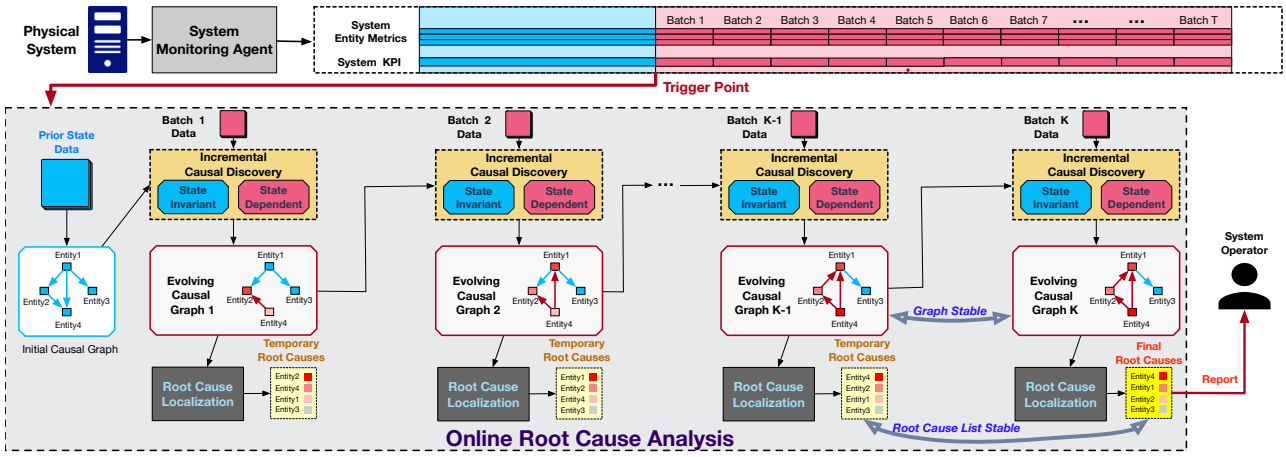


Figure 2: The overview of the proposed framework CORAL. CORAL first detects trigger points using the monitoring entity metrics and KPI data. If detected, it will initiate incremental causal graph learning. Each data batch is used to incrementally update the previous causal graph by disentangling the state-invariant and state-dependent causations for RCA. When the learned causal graph and the root cause list converge, system operators will receive the final root causes for system recovery.

entities with anomalous metrics can be the root causes of abnormal system latency/connection time, which is a sign of a system fault.

Trigger Point or System State Change Point is the time when the system transitions from one state to another. Real-world systems are dynamic. A system fault can cause a change in the system's status. As the state of a system varies, the underlying causal relationships between its components also change. Thus, to effectively identify root causes in an online setting, it is essential to learn different causal graphs in different states. From this perspective, the system state change points can be viewed as the triggers for updating the causal graph/online RCA model.

Problem Statement. Let $\mathcal{X} = \{\mathbf{X}_1, \cdots, \mathbf{X}_N\}$ denote N multivariate metric data. The i-th metric data is $\mathbf{X}_i = [\mathbf{x}_1^i, \cdots, \mathbf{x}_T^i]$, where $\mathbf{x}_t^i \in \mathbb{R}^M$ is the observation of M system entities at time point t. To reduce notational clutter, we omit the metric index i and use $\mathbf{X}$ to represent $\mathbf{X}_i$ in the following sections. These observations are non-stationary, and the relationships between various system entities are dynamic and subject to change over time. We assume that the underlying state of the system can change when a system fault occurs, and the relationships between system entities in each state are represented by a directed acyclic graph (DAG). For simplicity, we illustrate here using the system state transition from s_p to s_{p+1}. The system KPI is $\mathbf{y}$. The monitoring entity metric data of s_p is $\tilde{\mathbf{X}}_p \in \mathbb{R}^{\rho \times M}$, where ρ is the time length in the state s_p. G_p represents the causal graph of s_p, which consists of nodes representing system KPI or entities, and edges representing causal relations. The data of state s_{p+1} comes one batch at a time, denoted by $\tilde{\mathbf{X}}_{p+1} = [\check{\mathbf{X}}_{p+1}^1, \cdots, \check{\mathbf{X}}_{p+1}^L]$, where the l-th batch $\check{\mathbf{X}}_{p+1}^l \in \mathbb{R}^{b \times M}$ and b is the length of each batch of data. Our goal is to automatically trigger the RCA process when a system fault occurs, incrementally update G_p to G_{p+1} by considering each batch of data sequentially, and efficiently identify the top K nodes in the causal graph G_{p+1} that are most relevant to $\mathbf{y}$.

3 METHODOLOGY

Fig. 2 illustrates the overview of the proposed framework CORAL for online root cause localization.

3.1 Trigger Point Detection

Our goal is to detect the trigger points by integrating both entity metrics and system KPI data. Inspired by [2], we model the underlying dynamics of system entity metrics and KPI data (*i.e.*, multivariate time series observations) through the Multivariate Singular Spectrum Analysis (MSSA) model. For simplicity, we add the system KPI, $\mathbf{y}$, as one dimension to the metric data, $\mathbf{X}$, to illustrate our model.

Specifically, given monitoring metric data $\mathbf{X}$, we first construct the base matrix, denoted by $\mathbf{Z}_\mathbf{X}$, by using the previous T_0 records. The requirement for the initial value of T_0 here is that no system state transition occurs in the time segment $t \le T_0$. The singular vectors of $\mathbf{Z}_\mathbf{X}$ are then grouped into two matrices $\hat{\mathbf{U}}_0$ and $\hat{\mathbf{U}}_\perp$. We estimate the pre-change subspace $\hat{\mathbb{L}}_0$ as follows:

$$\hat{\mathbb{L}}_0 = \text{span}(\hat{\mathbf{U}}_0) \tag{1}$$

Meanwhile, let $\hat{\mathbb{L}}_\perp = \text{span}(\hat{\mathbf{U}}_\perp)$ be the orthogonal complement of the subspace $\hat{\mathbb{L}}_0$. After that, for the new data $t > T_0$, we build the L-lagged matrix $\mathbf{X}(t-L+1:t)$. We compute the Euclidean distance between the L-lagged matrix and the estimated subspace $\hat{\mathbb{L}}_0$ as the detection score, which can be defined as:

$$D(t) = \left\| \hat{\mathbf{U}}_\perp^\top \mathbf{X}(t-L+1:t) \right\|_F^2 - c \tag{2}$$

where $c \ge 0$ is the shift-downwards constant. Moreover, we compute the cumulative detection score using the cumulative sum (CUSUM) statistics, which can be defined as:

$$y(t) = \max\{y(t-1) + D(t), 0\} \quad y(T_0) = 0 \tag{3}$$

if the $y(t) = 0$, we proceed to check the next time point. Otherwise, the change point is identified when $y(t) > h$, where h is a predefined threshold. This process can be defined as:

$$\hat{\tau} = \inf_{t > T_0} \{t | y(t) \ge h\} \tag{4}$$

The change point $\hat{\tau}$ is the trigger for incremental causal graph learning. To recheck the next change point, the base matrix is updated with the time segment $\mathbf{X}(\hat{\tau}, \hat{\tau} + T_0 - 1)$. This model can detect trigger points in nearly real-time.

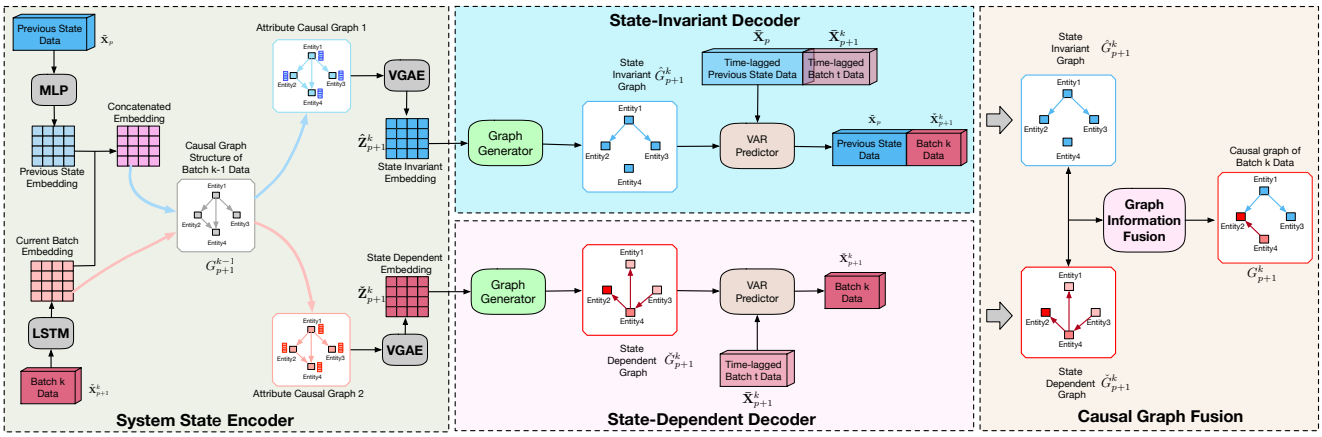


Figure 3: Incremental Disentangled Causal Graph Learning. The system state encoder aims to decouple state-invariant and state-dependent information in order to produce corresponding embeddings. State-invariant and state-dependent decoders aim to reconstruct and refine state-invariant and state-dependent causal relations, respectively. The causal graph fusion module aims to fuse state-invariant and state-dependent causation in order to obtain the new causal graph.

3.2 Incremental Disentangled Causal Graph Learning

After a trigger point is detected, we propose a disentangled causal graph learning model by integrating state invariant and state-dependent information, as well as incremental learning.

System State Encoder. The goal of the system state encoder is to aggregate the information from system state data and the corresponding causal graph. We use the k-th batch data to illustrate our design. For simplicity, we assume that the previous state data $\tilde{\mathbf{X}}_p$ and the batch of new state data $\check{\mathbf{X}}^k_{p+1}$ both have the system KPI $\mathbf{y}$ as one dimension.

Given $\tilde{\mathbf{X}}_p$, $\check{\mathbf{X}}^k_{p+1}$ and the causal graph at the $k-1$ batch G^{k-1}_{p+1}, we aim to integrate their information into state-invariant and state-dependent embeddings, respectively. First, we employ a fully-connected linear layer to preserve the information of $\tilde{\mathbf{X}}_p$ into a latent representation $\mathbf{U}_p$, which can be defined as:

$$\mathbf{U}_p = \tilde{\mathbf{X}}_p \cdot \mathbf{W}_p + \mathbf{b}_p \tag{5}$$

where $\mathbf{W}_p$ and $\mathbf{b}_p$ are the weight matrix and bias item of the linear layer, respectively.

Then, to track the information change in the state $p+1$'s data batch, we employ a recurrent function $f(\cdot,\cdot)$. The function $f(\cdot,\cdot)$ takes $\check{\mathbf{X}}^k_{p+1}$ and the previous hidden state $\mathbf{H}^{k-1}_{p+1}$ as inputs and outputs a latent representation $\mathbf{H}^k_{p+1}$, which is defined as:

$$\mathbf{H}^k_{p+1} = f(\check{\mathbf{X}}^k_{p+1}, \mathbf{H}^{k-1}_{p+1}) \tag{6}$$

We use a long short-term memory network (LSTM) [23] to implement $f(\cdot,\cdot)$. Due to the computational overhead, we do not use the passed batch of data to update the new causal graph. Thus, it is effective to track the causal dynamics using the LSTM module.

In addition, because state-invariant causal relations would be affected by both the previous state data and the new batch of data, whereas state-dependent causal relations are only affected by the new batch of data, to obtain the state-invariant embedding $\hat{\mathbf{Z}}^k_{p+1}$, we first concatenate $\mathbf{U}_p$ and $\mathbf{H}^k_{p+1}$ together, and then map it to the causal graph G^{k-1}_{p+1} as the node embeddings. After that, we employ the mapping function $g(\cdot,\cdot)$ to convert the attributed graph into the state-invariant embedding, which is defined as:

$$\hat{\mathbf{Z}}^k_{p+1} = g(\mathbf{A}^{k-1}_{p+1}, \text{Concat}(\mathbf{U}_p, \mathbf{H}^k_{p+1})) \tag{7}$$

where $\mathbf{A}^{k-1}_{p+1}$ represents the adjacency matrix of G^{k-1}_{p+1} and Concat represents the concatenated operation. to implement the function $g(.)$, we employ a variational graph autoencoder (VGAE) [29]. VGAE embeds all information into an embedding space that is smooth and continuous. This is helpful for capturing the causal dynamics between different system entities. To obtain the state-dependent embedding $\check{\mathbf{Z}}^k_{p+1}$, we only map $\mathbf{H}^k_{p+1}$ to G^{k-1}_{p+1} as its attributes, and then employ another VGAE layer to convert the attributed graph to the state-dependent embedding. This process can be defined as:

$$\check{\mathbf{Z}}^k_{p+1} = g(\mathbf{A}^{k-1}_{p+1}, \mathbf{H}^k_{p+1}) \tag{8}$$

State-Invariant Decoder. The goal of the state-invariant decoder is to learn the invariant causal relations across two system states. To recover the state-invariant part, we first feed $\hat{\mathbf{Z}}^k_{p+1}$ into the graph generator layer to generate the corresponding state-invariant graph $\hat{G}^k_{p+1}$, which is defined as:

$$\hat{G}^k_{p+1} = \text{Sigmoid}(\hat{\mathbf{Z}}^k_{p+1} \cdot \hat{\mathbf{Z}}^{k\top}_{p+1}) \tag{9}$$

where Sigmoid is an activation function and $(.)^\top$ is a transpose operation. But since this process constructs the graph using the state-invariant embeddings only, it can't guarantee that the state-invariant causal relationships are shown accurately in this graph. To overcome this issue, two optimization objectives must be met: 1) Making $\hat{G}^k_{p+1}$ as similar to the previous causal graph G^{k-1}_{p+1} as possible. 2) Fitting $\hat{G}^k_{p+1}$ to both the previous and new state data batches. To achieve the first objective, we minimize the reconstruction loss $\mathcal{L}_{\hat{G}}$, which is defined as:

$$\mathcal{L}_{\hat{G}} = \left\| \hat{\mathbf{A}}^k_{p+1} - \mathbf{A}^{k-1}_{p+1} \right\|^2 \tag{10}$$

where $\hat{\mathbf{A}}^k_{p+1}$ and $\mathbf{A}^{k-1}_{p+1}$ are the adjacency matrices of $\hat{G}^k_{p+1}$ and G^{k-1}_{p+1}, respectively. To achieve the second objective, we fit the graph and data with a structural vector autoregressive (SVAR) model [41]. More specifically, given the time-lagged data of the previous state $\bar{\mathbf{X}}_p$ and the current new batch $\bar{\mathbf{X}}^k_{p+1}$, the SVAR-based predictive equations can be defined as:

$$\begin{cases} \tilde{\mathbf{X}}_p = \tilde{\mathbf{X}}_p \cdot \hat{\mathbf{A}}^k_{p+1} + \bar{\mathbf{X}}_p \cdot \hat{\mathbf{D}}^k_{p+1} + \tilde{\epsilon}_p \\ \check{\mathbf{X}}^k_{p+1} = \check{\mathbf{X}}^k_{p+1} \cdot \hat{\mathbf{A}}^k_{p+1} + \bar{\mathbf{X}}^k_{p+1} \cdot \hat{\mathbf{D}}^k_{p+1} + \check{\epsilon}^k_{p+1} \end{cases} \tag{11}$$

where $\tilde{\epsilon}_p$ and $\check{\epsilon}^k_{p+1}$ are vectors of centered error variables; $\hat{\mathbf{A}}^k_{p+1}$ is used to capture causal relations among system entities; and the weight matrix $\hat{\mathbf{D}}^k_{p+1}$ is used to model the contribution of time-lagged data for the predictive task. $\hat{\mathbf{A}}^k_{p+1}$ and $\hat{\mathbf{D}}^k_{p+1}$ are used to predict both the past state data and the current batch of data. To ensure the accuracy of learned causal structures, we minimize two predictive errors $\mathcal{L}_{\tilde{p}}$ and $\mathcal{L}_{\hat{p}}$, which are defined as:

$$\begin{cases} \mathcal{L}_{\tilde{p}} = \left\| \tilde{\mathbf{X}}_p - (\tilde{\mathbf{X}}_p \cdot \hat{\mathbf{A}}^k_{p+1} + \bar{\mathbf{X}}_p \cdot \hat{\mathbf{D}}^k_{p+1}) \right\|^2 \\ \mathcal{L}_{\hat{p}} = \left\| \check{\mathbf{X}}^k_{p+1} - (\check{\mathbf{X}}^k_{p+1} \cdot \hat{\mathbf{A}}^k_{p+1} + \bar{\mathbf{X}}^k_{p+1} \cdot \hat{\mathbf{D}}^k_{p+1}) \right\|^2 \end{cases} \tag{12}$$

State-Dependent Decoder The goal of the state-dependent decoder is to learn the new causal relations introduced by the new batch of data. Similar to the learning process of the state-invariant decoder, we first generate the state-dependent graph $\check{G}^k_{p+1}$ by applying the same strategy on the embedding $\check{\mathbf{Z}}^k_{p+1}$, which is defined as:

$$\check{G}^k_{p+1} = \text{Sigmoid}(\check{\mathbf{Z}}^k_{p+1} \cdot \check{\mathbf{Z}}^{k^\top}_{p+1}) \tag{13}$$

To ensure the causal graph $\check{G}^k_{p+1}$ is brought by the new batch of data $\check{\mathbf{X}}^k_{p+1}$, two optimization objectives must be met: 1) Making $\check{G}^k_{p+1}$ as similar to the complement of the previous causal graph as possible; 2) Fitting $\check{G}^k_{p+1}$ to the new batch of data. To achieve the first objective, we minimize the reconstruction loss $\mathcal{L}_{\check{G}}$, which is defined as:

$$\mathcal{L}_{\check{G}} = \left\| \check{\mathbf{A}}^k_{p+1} - (\sim \mathbf{A}^{k-1}_{p+1}) \right\|^2 \tag{14}$$

where $(\sim \mathbf{A}^{k-1}_{p+1})$ refers to the inversion of each element in the adjacency matrix $\mathbf{A}^{k-1}_{p+1}$ and $\check{\mathbf{A}}^k_{p+1}$ is the adjacency matrix of $\check{G}^k_{p+1}$. To achieve the second objective, we define a predictive equation using SVAR:

$$\check{\mathbf{Y}}^k_{p+1} = \check{\mathbf{X}}^k_{p+1} \cdot \check{\mathbf{A}}^k_{p+1} + \bar{\mathbf{X}}^k_{p+1} \cdot \check{\mathbf{D}}^k_{p+1} + \epsilon \tag{15}$$

where $\check{\mathbf{A}}^k_{p+1}$ captures the new causal relations introduced by the new data batch $\check{\mathbf{X}}^k_{p+1}$; and $\check{\mathbf{D}}^k_{p+1}$ is to model the contribution of time-lagged data for prediction. To ensure the accuracy of learned causal structures, we minimize the predictive error $\mathcal{L}_{\check{p}}$, which is defined as:

$$\mathcal{L}_{\check{p}} = \left\| \check{\mathbf{X}}^k_{p+1} - (\check{\mathbf{X}}^k_{p+1} \cdot \check{\mathbf{A}}^k_{p+1} + \bar{\mathbf{X}}^k_{p+1} \cdot \check{\mathbf{D}}^k_{p+1}) \right\|^2 \tag{16}$$

Causal Graph Fusion. From the state-invariant decoder and state-dependent decoder, we can obtain the state-invariant causal graph $\hat{G}^k_{p+1}$ and the state-dependent causal graph $\check{G}^k_{p+1}$, respectively. To generate the causal graph G^k_{p+1} for the current batch of data, simple addition will not work because it may result in dense and cyclical graphs. Here, we propose a new graph fusion layer to fuse the two causal graphs, which can be formulated as follows:

$$\mathbf{A}^k_{p+1} = \text{RELU}(\tanh(\hat{\mathbf{A}}^k_{p+1} \cdot \check{\mathbf{A}}^{k^\top}_{p+1} - \check{\mathbf{A}}^k_{p+1} \cdot \hat{\mathbf{A}}^{k^\top}_{p+1})) \tag{17}$$

where $\mathbf{A}^k_{p+1}$ is the adjacency matrix of G^k_{p+1}. The subtraction term, tanh, and RELU activation functions may regularize the adjacency matrix so that if the element in $\mathbf{A}^k_{p+1}$ is positive, its diagonal counterpart element will be zero. To strictly force the G^k_{p+1} to be unidirectional and acyclic, we adopt the following exponential trace function as constraints inspired by [56]:

$$h(\mathbf{A}^k_{p+1}) = tr(e^{\mathbf{A}^k_{p+1} \circ \mathbf{A}^k_{p+1}}) - M \tag{18}$$

where $\circ$ is the Hadamard product of two matrices and M is the number of nodes (*i.e.*, system entities). This function satisfies $h(\mathbf{A}^k_{p+1}) = 0$ if and only if $\mathbf{A}^k_{p+1}$ is acyclic.

Optimization. To generate a robust casual graph for the new data batch, we jointly optimize all the preceding loss functions. Thus, the final optimization objective is defined as:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\hat{G}} + \mathcal{L}_{\check{G}} + \mathcal{L}_{\tilde{p}} + \mathcal{L}_{\hat{p}} + \mathcal{L}_{\check{p}} \\ &+ \lambda_1 \cdot (\left\| \hat{\mathbf{A}}^k_{p+1} \right\|_1 + \left\| \check{\mathbf{A}}^k_{p+1} \right\|_1) + \lambda_2 \cdot h(\mathbf{A}^k_{p+1}) \end{aligned} \tag{19}$$

where $\|\cdot\|_1$ is the $L1$-norm, which is used to increase the sparsity of $\hat{G}^k_{p+1}$ and $\check{G}^k_{p+1}$to reduce the computational cost; λ_1 and λ_2 control the penalized degree of regularization items.

Model Convergence. The discovered causal structure and the associated root cause list may gradually converge as the number of new data batches increases. So we incorporate them as an indicator to automatically terminate the online RCA to avoid unnecessary computing resource waste. Assume two consecutive causal graphs are G^{K-1}_{p+1}, G^K_{p+1}, and the associated root cause lists are $\mathbf{l}^{K-1}_{p+1}$, $\mathbf{l}^K_{p+1}$.

The node set of the two causal graphs is fixed. The edge distribution should be comparable when the causal graph converges. Thus, we define the graph similarity ς_G using the Jensen-Shannon divergence [17] as follows:

$$\varsigma_G = 1 - \text{JS}(P(G^{K-1}_{p+1}) || P(G^K_{p+1})) \tag{20}$$

where $P(.)$ refers to the edge distribution of the corresponding graph. ς_G has a value range of $[0 \sim 1]$. The greater the value of ς_G is, the closer the two graphs are.

We use the rank-biased overlap metric [55] (RBO) to calculate the similarity between two root cause lists in order to fully account for the changing trend of root cause rank. The ranked list similarity $\varsigma_{\mathbf{l}}$ is defined as:

$$\varsigma_{\mathbf{l}} = RBO(\mathbf{l}^{K-1}_{p+1}, \mathbf{l}^K_{p+1}) \tag{21}$$

$\varsigma_{\mathbf{l}}$ has a value range of $[0 \sim 1]$. The greater the value of $\varsigma_{\mathbf{l}}$ is, the more similar the two root cause lists are. We use a hyperparameter $\alpha \in [0 \sim 1]$ to integrate ς_G and $\varsigma_{\mathbf{l}}$, defined as:

$$\varsigma = \alpha \cdot \varsigma_G + (1 - \alpha) \cdot \varsigma_{\mathbf{l}} \tag{22}$$

The online RCA process may stop when ς is more than a threshold.

3.3 Network Propagation based Root Cause Localization

After obtaining the causal graph G^k_{p+1}, there are two kinds of nodes: system entities and KPI in the graph. However, the system entities linked to the KPI may not always be the root causes. This is because the malfunctioning effects will spread to neighboring entities starting from the root causes [11, 13, 33]. We present a random walk-based method for capturing such patterns and more precisely locating root causes. For simplicity, in this subsection, we directly use G to represent G^k_{p+1}. To trace back the root causes, we first transpose the learned causal graph to get $G^\top$, then adopt a random walk with restarts on the transposed causal graph to estimate the probability score of each entity by starting from the KPI node.

Specifically, we assume that the transition probabilities of a particle on the transposed structure may be represented by $\mathbf{H}$, which has the same shape as the adjacency matrix of $G^\top$. Each element in $\mathbf{H}$ indicates the transition probability between any two nodes. Imagine that from the KPI node, a particle begins to visit the causal structure. It jumps to any one node with a probability value $\phi \in [0, 1]$ or stays at the original position with $1 - \phi$. The higher the value of ϕ is, the more possible the jumping behavior happens. Specifically, if the particle moves from node i to node j, the moving probability in H should be updated by:

$$\mathbf{H}[i, j] = (1 - \phi)\mathbf{A}^\top[i, j] / \sum_{\kappa=1}^{M} \mathbf{A}^\top[i, \kappa] \tag{23}$$

where $\mathbf{A}^\top$ is the adjacency matrix of $G^\top$. During the visiting exploration process, we may restart from the KPI node to revisit other entities with the probability $\varphi \in [0, 1]$. Thus, the visiting probability transition equation of the random walk with restarts can be formulated as:

$$\mathbf{q}_{\tau+1} = (1 - \varphi) \cdot \mathbf{q}_\tau + \varphi \cdot \mathbf{q}_\xi \tag{24}$$

where $\mathbf{q}_\tau$ and $\mathbf{q}_{\tau+1}$ are the visiting probability distributions at the τ and $\tau + 1$ steps, respectively. $\mathbf{q}_\xi$ is the initial visiting probability distribution at the initial step. When the visiting probability distribution converges, the probability scores of the nodes are used as their causal scores to rank them. The top-K ranked nodes are the most likely root causes of the associated system fault.

4 EXPERIMENTS

4.1 Experimental Setup

4.1.1 Datasets. We conducted extensive experiments on three real-world datasets: **1) AIOps**: was collected from a real micro-service system that consists of 5 servers with 234 pods. From May 2021 to December 2021, the operators collected metric data (*e.g.*, CPU utilization or memory consumption) of all system entities. During this time period, there are 5 system faults. **2) SWaT** [37]: was collected over an 11-day period from a water treatment testbed that was equipped with 51 sensors. The system had been operating normally for the first 7 days before being attacked in the last 4 days. There are 16 system faults in the collection period. **3) WADI** [1]: was collected from a water treatment testbed over the operation of 16 days. The testbed consists of 123 sensors/actuators. The system had been operating normally for the first 14 days before being attacked in the last 2 days. There are 15 system faults in the collection period.

4.1.2 Evaluation Metrics. We compared CORAL with the following four widely-used metrics [20, 34, 38]: 1) **Precision@K (PR@K)**. It denotes the probability that the top-K predicted root causes are real, defined as: PR@K = $\frac{1}{|\mathbb{A}|}\sum_{a\in\mathbb{A}} \frac{\sum_{i<K} R_a(i)\in V_a}{min(K,|V_a|)}$, where $\mathbb{A}$ is the set of system faults; a is one fault in $\mathbb{A}$; V_a is the real root causes of a; R_a is the predicted root causes of a; and i refers to the i-th predicted cause of R_a. 2) **Mean Average Precision@K (MAP@K)**. It assesses the model performance in the top-K predicted causes from the overall perspective, defined as: MAP@K = $\frac{1}{K|\mathbb{A}|}\sum_{a\in\mathbb{A}}\sum_{1\le j\le K}$ PR@j, where a higher value indicates better performance. 3) **Mean Reciprocal Rank (MRR)**. This metric measures the ranking capability of models. The larger the MRR value is, the further ahead the predicted positions of the root causes are; thus, operators can find the real root causes more easily. MRR is defined as: MRR = $\frac{1}{\mathbb{A}}\sum_{a\in\mathbb{A}}\frac{1}{rank_{R_a}}$, where $rank_{R_a}$ is the rank number of the first correctly predicted root cause for system fault a. 4) **Ranking Percentile (RP).** Since CORAL is an online root cause localization framework, we expect the ranks of ground-truth root causes to rise as the learning process progresses. Here, we propose the ranking percentile to evaluate the learning process, defined as: RP = $(1 - \frac{rank_{R_a}}{N}) \times 100\%, \quad s.t. \quad a \in \mathbb{A}$, where N is the total number of nodes (*i.e.*, system entities).

4.1.3 Baselines. We compared CORAL with the following five causal discovery models: 1) **NOTEARS** [56] formulates the structure learning problem as a purely continuous constrained optimization problem over real matrices to increase learning efficiency. 2) **GOLEM** [40] employs a likelihood-based score function to relax the hard DAG constraint in NOTEARS. 3) **Dynotears** [41] is a score-based method that uses the structural vector autoregression model to construct dynamic Bayesian networks. 4) **PC** [47] is a classic constraint-based method. It first identifies the skeleton of the causal graph with the independence test, then generates the orientation direction using the v-structure and acyclicity constraints. 5) **C-LSTM** [50] captures the nonlinear Granger causality that existed in multivariate time series by using LSTM neural networks.

All these models can only learn the causal structure from time series data offline. Thus, we first collect monitoring data from the beginning until system failures occur as historical records. Then, based on the collected records, we apply the causal discovery models to learn causal graphs and leverage network propagation on such graphs to identify the top K nodes as the root causes. Besides, since CORAL is an online RCA framework, we extend NOTEARS and GOLEM to the online learning setting, denoted by **NOTEARS*** and **GOLEM***, respectively, for a fair comparison[1]. For the online setting, we collect monitoring data before the first trigger point as training or historical data to construct the initial causal graph, which describes the normal system state. Once a trigger point is detected, we update the causal graph iteratively for each new batch of data. CORAL can inherit the causations from the previous data batch. NOTEARS* and GOLEM* have to learn from scratch for each new data batch. All causal discovery baseline models are implemented using the gcastle library[2].

4.1.4 Experimental Settings. All experiments are conducted on a server running Ubuntu 18.04.5 with Intel(R) Xeon(R) Silver 4110

[1] Other baselines are not extended to the online setting as they are time-intensive when there are multiple data batches.
[2] https://github.com/huawei-noah/trustworthyAI/tree/master/gcastle.

Table 1: Overall performance *w.r.t.* SWaT dataset.

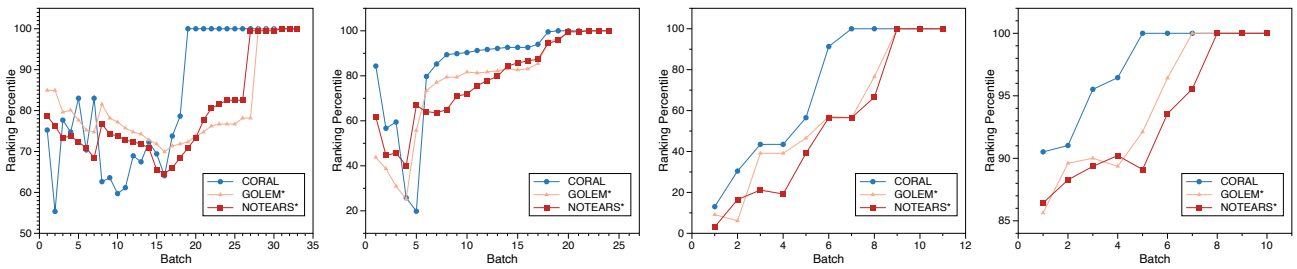
	PR@1	PR@3	PR@5	PR@7	PR@10	MAP@3	MAP@5	MAP@7	MAP@10	MRR
CORAL	6.25%	**31.25%**	**55.21%**	**64.58%**	**92.71%**	**15.63%**	**29.79%**	**39.73%**	**53.96%**	31.72%
NOTEARS	6.25%	7.29%	12.50%	39.58%	47.92%	7.64%	9.58%	16.96%	25.00%	22.36%
GOLEM	**18.75%**	7.29%	18.75%	54.17%	62.50%	11.81%	13.33%	22.02%	33.44%	30.42%
Dynotears	**18.75%**	25.00%	29.17%	41.67%	58.33%	23.96%	26.04%	29.17%	37.08%	**33.99%**
PC	12.50%	21.88%	36.46%	47.92%	53.13%	19.79%	26.04%	31.40%	37.40%	32.27%
C-LSTM	12.50%	27.08%	27.08%	39.58%	60.42%	19.44%	22.50%	26.49%	34.17%	32.86%
NOTEARS*	6.25%	29.17%	36.46%	55.21%	67.71%	14.93%	23.54%	32.59%	42.19%	26.30%
GOLEM*	6.25%	29.17%	42.71%	57.29%	68.75%	17.01%	26.04%	34.97%	43.65%	28.09%

Table 2: Overall performance *w.r.t.* WADI dataset.

	PR@1	PR@3	PR@5	PR@7	PR@10	MAP@3	MAP@5	MAP@7	MAP@10	MRR
CORAL	**35.71%**	**23.81%**	**60.00%**	**70.24%**	**83.33%**	**28.71%**	**36.05%**	**45.82%**	**56.00%**	**51.90%**
NOTEARS	7.14%	23.81%	30.00%	35.71%	41.67%	17.46%	22.55%	26.14%	30.80%	30.12%
GOLEM	7.14%	10.71%	40.00%	51.19%	64.29%	9.52%	20.14%	28.33%	38.05%	25.89%
Dynotears	14.29%	29.76%	32.86%	42.86%	46.43%	20.63%	25.38%	29.35%	33.76%	34.28%
PC	14.29%	21.43%	35.71%	45.24%	57.14%	16.67%	24.29%	28.91%	35.71%	30.74%
C-LSTM	12.50%	21.43%	46.43%	52.38%	64.29%	17.86%	27.86%	34.69%	42.62%	33.28%
NOTEARS*	14.29%	20.24%	45.71%	66.67%	72.62%	18.65%	27.48%	38.67%	48.38%	37.74%
GOLEM*	21.43%	20.24%	60.00%	64.29%	73.81%	19.84%	30.33%	39.86%	48.98%	40.24%

Table 3: Overall performance *w.r.t.* AIOps dataset.

	PR@1	PR@3	PR@5	PR@7	PR@10	MAP@3	MAP@5	MAP@7	MAP@10	MRR
CORAL	**80.00%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**93.33%**	**96.00%**	**97.14%**	**98.00%**	**90.00%**
NOTEARS	0.00%	40.00%	80.00%	40.00%	60.00%	20.00%	28.00%	37.14%	44.00%	20.46%
GOLEM	20.00%	60.00%	60.00%	60.00%	60.00%	40.00%	40.00%	51.43%	54.00%	37.74%
Dynotears	40.00%	60.00%	60.00%	60.00%	60.00%	53.33%	56.00%	57.14%	58.00%	50.77%
PC	20.00%	20.00%	20.00%	40.00%	60.00%	20.00%	20.00%	22.86%	30.00%	25.36%
C-LSTM	0.00%	40.00%	60.00%	60.00%	60.00%	26.67%	36.00%	42.86%	48.00%	24.73%
NOTEARS*	40.00%	80.00%	80.00%	80.00%	80.00%	66.67%	72.00%	74.29%	76.00%	60.00%
GOLEM*	60.00%	80.00%	80.00%	80.00%	80.00%	73.33%	76.00%	77.14%	78.00%	70.00%



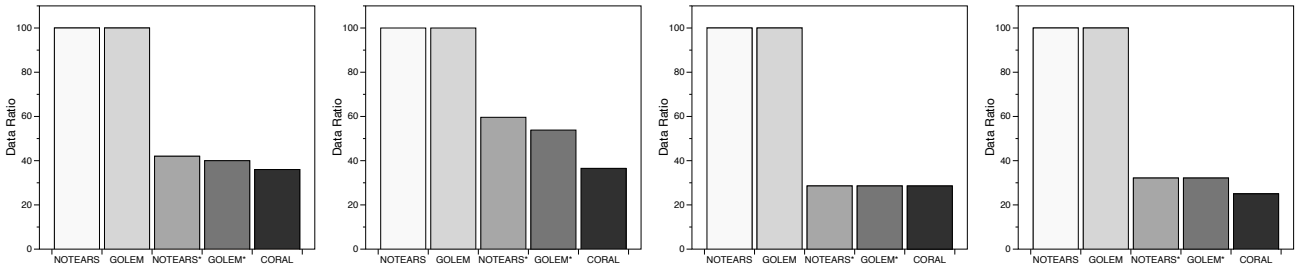
(a) AIOps 0901 (b) AIOps 0524 (c) SWaT Fault 8 (d) WADI Fault 6

Figure 4: Comparison of online RCA models on different batches in terms of ranking percentile.

CPU @ 2.10GHz, 4-way GeForce RTX 2080 Ti GPUs, and 192 GB memory. In addition, all methods were implemented using Python 3.8.12 and PyTorch 1.7.1.

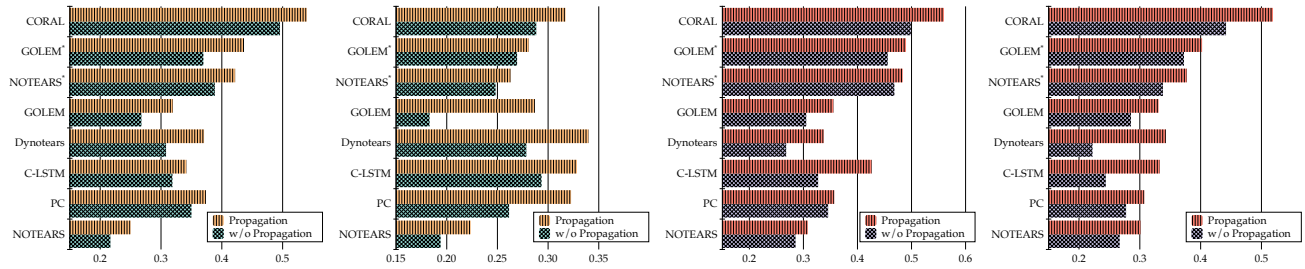
4.2 Performance Evaluation

4.2.1 Overall Performance. Table 1, Table 2, and Table 3 show the overall performance of all models. The greater the value of evaluation measures, the superior the model's performance. There are two key observations: First, the online methods (*e.g.*, NOTEARS*, GOLEM*, and CORAL) outperform the offline approaches (*e.g.*, Dynotears, PC, and C-LSTM). The underlying driver is that online methods can rapidly capture the changing patterns in the monitoring metric data. So, they can learn an accurate and noise-free causal structure for RCA. Second, compared to online methods, CORAL still significantly outperforms NOTEARS* and GOLEM*. The underlying reason is that the disentangled causal graph learning module independently learns state-invariant and state-dependent causal relations for incrementally updating the causal graph. Such a learning manner can learn more robust and accurate causal relations, enhancing root localization performance. Thus, the experimental results demonstrate the superiority and effectiveness of CORAL in root cause localization over other baselines.



(a) AIOps 0901 (b) AIOps 0524 (c) SWaT Fault 8 (d) WADI Fault 6

Figure 5: Comparison of required data volume for different root cause analysis models.



(a) SWaT MAP@10 (b) SWaT MRR (c) WADI MAP@10 (d) WADI MRR

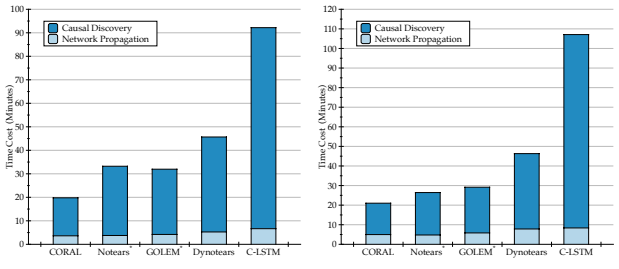
Figure 6: Study of the impact of network propagation for RCA.

4.2.2 Learning Procedure Analysis. Fig. 4 illustrates how the performance of online RCA frameworks varies as the number of data batches grows. As more data comes in, we can see that the ranking percentiles of root causes identified by all online frameworks can converge. This shows how well the online RCA works, since it can gradually pick up on the changing patterns in monitoring metric data. Another interesting observation is that CORAL can identify the root causes earlier than NOTEARS*, GOLEM*. For instance, in the AIOps 0901 case, CORAL can identify the root causes nine batches earlier than the other two models. A possible reason is that CORAL updates the new causal graph by disentangling the state-invariant and state-dependent information instead of learning from scratch. It may generate more robust and effective causal structures. This experiment shows the validity of RCA's online learning setup and the utility of disentangled causal graph learning.

4.2.3 Required Data Volume Analysis. Fig. 5 shows the comparison results of required data volumes for different RCA models. We find that compared with offline models, the online RCA approaches can significantly reduce the required data volume. In AIOps 0524, for instance, NOTEARS* reduces data usage by at least 40% compared to the offline models. A possible explanation is that the online RCA setting may capture the system state dynamics in order to acquire a good causal structure without being affected by an excessive amount of redundant data. Moreover, we observe that CORAL uses fewer data but achieves better results compared with other baselines. A potential reason is that disentangled causal graph learning may inherit causal information from the previous state, resulting in reduced data consumption while preserving a robust causal structure and good RCA performance. Thus, this experiment shows that CORAL can reduce the computational costs of RCA.

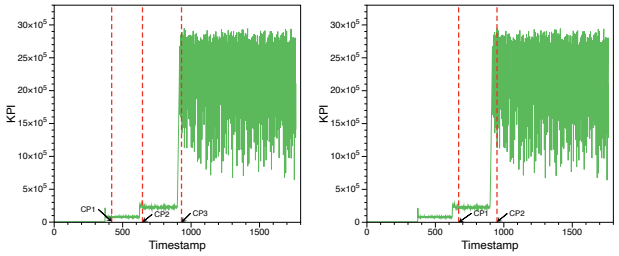
4.2.4 Influence of Network Propagation. Here, we apply our network propagation mechanism (see Section 3.3) to the causal structures learned by each causal discovery model to investigate its impact on performance. For the control group, we eliminate the propagation effect by directly selecting the system entities connected to the system KPI as root causes. Fig. 6 shows the comparison results of all models on SWaT and WADI data in terms of MAP@10 and MRR. We can find that network propagation significantly enhances the RCA performance in all cases. A possible reason for this observation is that in most cases, the malfunctioning effects of root causes may propagate among system entities over time. Network propagation can simulate such patterns in the learned causal structure, resulting in a better RCA performance. Thus, this experiment demonstrates that the network propagation module is critical to the performance of an RCA model.

4.2.5 Time Cost Analysis. Fig. 7 shows the time cost comparison of causal structure learning and network propagation of distinct models using two fault cases of AIOps, denoted by 0524 and 0901, respectively. We can observe that offline methods require more time than online methods for network propagation. A potential reason is that offline methods utilize a large amount of historical data for learning causal relations, resulting in a more dense causal network with noisy causal relations than online methods. Thus, it requires more time for the network propagation module of offline methods to converge. Another interesting observation is that CORAL requires the least amount of time in causal structure learning compared with other baselines. The underlying driver is that the incremental causal graph learning module in CORAL maintains the state-invariant causal relations and updates the state-dependent ones, hence accelerating the causal structure learning.



(a) AIOps 0524 (b) AIOps 0901

Figure 7: Comparison of the time costs associated with causal discovery and network propagation of distinct models.



(a) AIOps 0524 (Metric data + KPI) (b) AIOps 0524 (KPI)

Figure 8: Comparison of trigger point detection with and w/o metric data. Red dashed lines indicate the trigger points.

4.2.6 Trigger Point Detection Analysis. In our proposed trigger point detection module (see Section 3.1), we use both system metric data and KPI to identify the transition points between system states. To evaluate its effect on performance, we use only the system KPI data for determining the trigger points in the control group. Fig. 8 shows the comparison results on AIOps 0524 data by visualizing its system KPI and associated trigger points (highlighted by red dashed lines). We find that by integrating metric data into KPI, we can detect system transition points more quickly and precisely. A potential reason is that metric data may contain some early failure symptoms or precursor patterns, which can not be captured in the system key performance indicator.

5 RELATED WORK

Root Cause Analysis (RCA) focuses on identifying the root causes of system failures/faults based on symptom observations [46]. Many offline RCA approaches [12, 14, 19, 45, 49] have been proposed to improve system robustness in various domains. In energy domains, Capozzoli *et al.* utilized statistical techniques and DNNs to determine the cause of abnormal energy consumption [9]. In web development domains, Brandon *et al.* proposed a graph representation framework to identify root causes in microservice systems by comparing anomalous circumstances and graphs [8]. Different from existing offline studies, CORAL is an online RCA approach.

Causal Discovery in Time Series aims to discover causal structures using observational time series data [5]. Existing approaches are four-fold: (i) Granger causality approaches [39, 50], in which causality is assessed according to whether one time series is useful for predicting another; (ii) Constraint-based approaches [43, 48], in which causal skeleton is first identified using conditional independence test, and then the causal directions are determined based on the Markov equivalence relations; (iii) Noise-based approaches [25, 42], in which causal relations among distinct variables are depicted by the combinations between these variables and associated noises; (iv) Score-based approaches [7, 41, 52], in which the validity of a candidate causal graph is evaluated using score functions, and the graph with the highest score is the final output. Existing studies assume that the causal graph underlying the time series is constant and stable. However, real-world causal relationships between different variables are dynamic and vary over time. Our study proposes a novel incremental causal discovery model based on disentangled graph learning.

Change Point Detection is to identify the state transitions in time-series data. It can be categorized into offline detection and online detection [3]. Offline change point detection takes the entire historical time series as input and outputs all possible change points at once. It has many applications including speech processing [44], financial analysis [16], bio-informatics [35]. Online change point detection examines new data once available in order to detect new change points quickly. It can detect change points in real-time or near real-time when time series are monitored. The method is frequently used to detect the occurrence of major incidents/events (*e.g.*, system faults) in monitoring systems [6, 27]. To rapidly identify state transitions, we employ an online change point detection method using multivariate singular spectrum analysis [2].

Disentangled Representation Learning aims to identify and disentangle the underlying explanatory factors hidden in the observational data [53]. Disentangled representation Learning has been widely used in a variety of domains, including computer vision [24], time series analysis [10, 32], graph learning [30, 51, 54], and natural language processing [22]. In particular, for graph learning, Guo *et al.* [21] proposed an unsupervised disentangled approach to disentangle node and edge features from attributed graphs. To learn graph-level disentangled representations, Li *et al.* [30] presented a method for learning disentangled graph representations with self-supervision. Different from the previous works, we identify and disentangle the state-invariant and state-dependent factors for incremental causal graph learning.

6 CONCLUSION

In this paper, we studied a challenging problem of incremental causal structure learning for online RCA. To tackle it, we first designed an online trigger point detection module to detect system state changes and trigger early RCA. To efficiently learn causation among system entities, we proposed a novel incremental disentangled causal graph learning approach to incrementally update the causal graph. Based on the learned graph, we used a random walk with restarts to model the network propagation of system faults. We conducted comprehensive experiments on three real-world datasets to evaluate the proposed framework. The experimental results validate its effectiveness and the importance of each module in CORAL. An interesting direction for further exploration would be incorporating other sources of data, such as system logs, with the time series data for online root cause analysis in complex systems.

7 ACKNOWLEDGMENTS

This research was partially supported by the National Science Foundation (NSF) via grant numbers: 2040950, 2006889, and 2045567.

REFERENCES

[1] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. 2017. WADI: a water distribution testbed for research in the design of secure cyber physical systems. In *CySWater*. 25–28.

[2] Arwa Alanqary, Abdullah Alomar, and Devavrat Shah. 2021. Change Point Detection via Multivariate Singular Spectrum Analysis. *NeurIPS* 34 (2021), 23218–23230.

[3] Samaneh Aminikhanghahi and Diane J Cook. 2017. A survey of methods for time series change point detection. *Knowledge and Information Systems* 51, 2 (2017), 339–367.

[4] Bjørn Andersen and Tom Fagerhaug. 2006. *Root cause analysis: simplified tools and techniques*. Quality Press.

[5] Charles K Assaad, Emilie Devijver, and Eric Gaussier. 2022. Survey and Evaluation of Causal Discovery Methods for Time Series. *JAIR* 73 (2022), 767–819.

[6] Azzeddine Bakdi, Wahiba Bounoua, Amar Guichi, and Saad Mekhilef. 2021. Real-time fault detection in PV systems under MPPT using PMU and high-frequency multi-sensor data through online PCA-KDE-based multivariate KL divergence. *International Journal of Electrical Power & Energy Systems* 125 (2021), 106457.

[7] Alexis Bellot, Kim Branson, and Mihaela van der Schaar. 2021. Neural graphical modelling in continuous-time: consistency guarantees and algorithms. In *ICLR*.

[8] Álvaro Brandón, Marc Solé, Alberto Huélamo, David Solans, María S Pérez, and Victor Muntés-Mulero. 2020. Graph-based root cause analysis for service-oriented and microservice architectures. *Journal of Systems and Software* 159 (2020), 110432.

[9] Alfonso Capozzoli, Fiorella Lauro, and Imran Khan. 2015. Fault detection analysis using data mining techniques for a cluster of smart office buildings. *Expert Systems with Applications* 42, 9 (2015), 4324–4338.

[10] Zhengzhang Chen, Haifeng Chen, and Yuening Li. 2022. Interpretable time series representation learning with multiple-level disentanglement. US Patent App. 17/582,191.

[11] 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 *SIGKDD*. 805–814.

[12] Ailin Deng and Bryan Hooi. 2021. Graph neural network-based anomaly detection in multivariate time series. In *AAAI*, Vol. 35. 4027–4035.

[13] Boxiang Dong, Zhengzhang Chen, Hui Wang, Lu-An Tang, Kai Zhang, Ying Lin, Zhichun Li, and Haifeng Chen. 2017. Efficient discovery of abnormal event sequences in enterprise security systems. In *CIKM*. 707–715.

[14] George K Fourlas and George C Karras. 2021. A survey on fault diagnosis methods for UAVs. In *ICUAS*. IEEE, 394–403.

[15] Frank Bajak. 2021. Why did Amazon Web Services crash? Here's what it means. https://globalnews.ca/news/8434673/why-amazon-web-services-crash/.

[16] Klaus Frick, Axel Munk, and Hannes Sieling. 2014. Multiscale change point inference. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 76, 3 (2014), 495–580.

[17] Bent Fuglede and Flemming Topsoe. 2004. Jensen-Shannon divergence and Hilbert space embedding. In *ISIT*. 31.

[18] Alexander Gepperth and Barbara Hammer. 2016. Incremental learning algorithms and applications. In *ESANN*.

[19] Jiaping Gui, Ding Li, Zhengzhang Chen, Junghwan Rhee, Xusheng Xiao, Mu Zhang, Kangkook Jee, Zhichun Li, and Haifeng Chen. 2020. APTrace: A responsive system for agile enterprise level causality analysis. In *ICDE*. 1701–1712.

[20] Xiaoxiao Guo, Hui Wu, Yu Cheng, Steven Rennie, Gerald Tesauro, and Rogerio Feris. 2018. Dialog-based interactive image retrieval. *NeurIPS* 31 (2018).

[21] Xiaojie Guo, Liang Zhao, Zhao Qin, Lingfei Wu, Amarda Shehu, and Yanfang Ye. 2020. Interpretable deep graph generation with node-edge co-disentanglement. In *SIGKDD*. 1697–1707.

[22] Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *ACL*. 388–397.

[23] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[24] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. 2018. Learning to decompose and disentangle representations for video prediction. *NeurIPS* 31 (2018).

[25] Aapo Hyvärinen, Kun Zhang, Shohei Shimizu, and Patrik O Hoyer. 2010. Estimation of a structural vector autoregression model using non-gaussianity. *Journal of Machine Learning Research* 11, 5 (2010).

[26] Muhammad Azam Ikram, Sarthak Chakraborty, Subrata Mitra, Shiv Saini, Saurabh Bagchi, and Murat Kocaoglu. 2022. Root Cause Analysis of Failures in Microservices through Causal Discovery. In *NeurIPS*.

[27] Yuchen Jiao, Yanxi Chen, and Yuantao Gu. 2018. Subspace Change-Point Detection: A New Model and Solution. *IEEE Journal of Selected Topics in Signal Processing* 12, 6 (2018), 1224–1239.

[28] Emre Kıcıman and Lakshminarayanan Subramanian. 2005. Root cause localization in large scale systems. In *HotDep*.

[29] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[30] Haoyang Li, Xin Wang, Ziwei Zhang, Zehuan Yuan, Hang Li, and Wenwu Zhu. 2021. Disentangled contrastive learning on graphs. In *NeurIPS*, Vol. 34. 21872–21884.

[31] 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 *SIGKDD*. 3230–3240.

[32] Yuening Li, Zhengzhang Chen, Daochen Zha, Mengnan Du, Jingchao Ni, Denghui Zhang, Haifeng Chen, and Xia Hu. 2022. Towards Learning Disentangled Representations for Time Series. In *SIGKDD*. 3270–3278.

[33] Ying Lin, Zhengzhang Chen, Cheng Cao, Lu-An Tang, Kai Zhang, Wei Cheng, and Zhichun Li. 2018. Collaborative alert ranking for anomaly detection. In *CIKM*. 1987–1995.

[34] 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*. 338–347.

[35] Siqi Liu, Adam Wright, and Milos Hauskrecht. 2018. Change-point detection method for clinical decision support system rule monitoring. *Artificial Intelligence in Medicine* 91 (2018), 49–56.

[36] Chen Luo, Zhengzhang Chen, Lu-An Tang, Anshumali Shrivastava, Zhichun Li, Haifeng Chen, and Jieping Ye. 2018. TINET: learning invariant networks via knowledge transfer. In *SIGKDD*. 1890–1899.

[37] Aditya P Mathur and Nils Ole Tippenhauer. 2016. SWaT: A water treatment testbed for research and training on ICS security. In *CySWater*. 31–36.

[38] 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.

[39] Meike Nauta, Doina Bucur, and Christin Seifert. 2019. Causal discovery with attention-based convolutional neural networks. *Machine Learning and Knowledge Extraction* 1, 1 (2019), 312–340.

[40] Ignavier Ng, AmirEmad Ghassami, and Kun Zhang. 2020. On the role of sparsity and dag constraints for learning linear dags. *NeurIPS* 33 (2020), 17943–17954.

[41] Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. 2020. Dynotears: Structure learning from time-series data. In *AISTATS*. 1595–1605.

[42] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. 2013. Causal inference on time series using restricted structural equation models. *NeurIPS* 26 (2013).

[43] Jakob Runge. 2020. Discovering contemporaneous and lagged causal relations in autocorrelated nonlinear time series datasets. In *UAI*. 1388–1397.

[44] Nicolas Seichepine, Slim Essid, Cédric Févotte, and Olivier Cappé. 2014. Piecewise constant nonnegative matrix factorization. In *ICASSP*. IEEE, 6721–6725.

[45] Jacopo Soldani and Antonio Brogi. 2022. Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey. *ACM Computing Surveys (CSUR)* 55, 3 (2022), 1–39.

[46] Marc Solé, Victor Muntés-Mulero, Annie Ibrahim Rana, and Giovani Estrada. 2017. Survey on models and techniques for root-cause analysis. *arXiv preprint arXiv:1701.08546* (2017).

[47] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. 2000. *Causation, prediction, and search*. MIT press.

[48] Jie Sun, Dane Taylor, and Erik M Bollt. 2015. Causal network inference by optimal causation entropy. *SIAM Journal on Applied Dynamical Systems* 14, 1 (2015), 73–106.

[49] LuAn Tang, Hengtong Zhang, Zhengzhang Chen, Bo Zong, LI Zhichun, Guofei Jiang, and Kenji Yoshihira. 2019. Graph-based attack chain discovery in enterprise security systems. US Patent 10,289,841.

[50] A Tank, I Covert, N Foti, A Shojaie, and EB Fox. 2021. Neural Granger Causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[51] Dongjie Wang, Zhengzhang Chen, Yanjie Fu, Yanchi Liu, and Haifeng Chen. 2023. Incremental Causal Graph Learning for Online Unsupervised Root Cause Analysis. *arXiv preprint arXiv:2305.10638* (2023).

[52] Dongjie Wang, Zhengzhang Chen, Jingchao Ni, Liang Tong, Zheng Wang, Yanjie Fu, and Haifeng Chen. 2023. Hierarchical Graph Neural Networks for Causal Discovery and Root Cause Localization. *arXiv preprint arXiv:2302.01987* (2023).

[53] Xin Wang, Hong Chen, Si'ao Tang, Zihao Wu, and Wenwu Zhu. 2022. Disentangled Representation Learning.

[54] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In *SIGIR*. 1001–1010.

[55] William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)* 28, 4 (2010), 1–38.

[56] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. 2018. Dags with no tears: Continuous optimization for structure learning. *NeurIPS* 31 (2018).