Root Cause Analysis of Failures from Partial Causal Structures

Azam Ikram[†1] **Kenneth Lee**[†1] **Shubham Agarwal**[2] **Shiv Kumar Saini**[2] **Saurabh Bagchi**[1] **Murat Kocaoglu**[1]

[1]Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA
[2]Adobe Research, India

Abstract

Finding the root cause of failures is a prominent problem in many complex networks. Causal inference provides us with tools to address this problem algorithmically to automate this process and solve it efficiently. The existing methods either use a known causal structure to identify root cause by backtracking the changes, or ignore the causal structure but relies on invariance tests to identify the changing causal mechanisms after the failure. Assuming a single, unknown root cause, we first establish a novel connection between root cause analysis and the *Interactive Graph Search (IGS)* problem. This mapping highlights the importance of causal knowledge: we demonstrate that any algorithm relying solely on marginal invariance tests to identify the root cause must perform at least $\Omega(\log_2(n) + d\log_{1+d} n)$ many tests, where n represents the number of components and d denotes the maximum out-degree of the graph. We then present an optimal algorithm that achieves this bound by reducing the root cause identification problem as an instance of IGS. Beyond the single root cause scenario, we propose a practical extension for settings with multiple root causes and partial causal knowledge. More specifically, we show that even if the causal graph is partially known, we can identify the root-causes with a linear number of invariance tests. This is the first known result on incorporating a partial causal structure for root cause analysis. Our experiments on a production-level application demonstrate that, even in the absence of complete causal information, our approach accurately identifies the root causes of failures. Our source code is available online at github.com/azamikram/rcg.

[†]Equal contribution

1 INTRODUCTION

Root Cause Analysis (RCA), which aims to understand the root cause of failures, is crucial for ensuring the reliability and stability of production systems in diverse domains, including but not limited to medicine [Kellogg et al., 2016, Latino, 2015], telecommunications [Schaaf et al., 2015], and IT operations [Whitney and Daniels, 2013, Drasar and Jirsik, 2019]. In cloud applications, particularly those using microservice architectures, the challenges of RCA are even more pronounced. The large number of microservices complicates pinpointing the primary cause of failures [Emmons et al., 2022], and the interdependent nature of these services means that a failure in one can cascade, disrupting the entire network. These factors make timely and accurate diagnosis of failures particularly difficult. According to Wang et al. [2018], identifying the root cause of issues in platforms like IBM's Bluemix can take an average of three hours without automated tools. Therefore, rapid fault detection is essential for minimizing downtime and mitigating impact on system performance. Delays in diagnosing issues can lead to substantial financial losses and customer dissatisfaction, especially as service-level agreements often prioritize system availability as a key performance indicator.

Recent RCA research has focused on developing methods to detect the root cause of failures, often through a two-phase process: first, constructing a graph structure and then ranking the nodes within that graph. Some approaches rely on expert knowledge to build the graph [Ma et al., 2020], while others derive it from service call graphs [Chakraborty et al., 2023], or employ deep neural networks for graph learning [Lin et al., 2024]. The goal is to model relationships and dependencies between services using causal discovery techniques to construct a causal graph [Wang et al., 2018, Qiu et al., 2020, Gan et al., 2021, Ikram et al., 2022, Xin et al., 2023]. For instance, MicroCause [Meng et al., 2020] employs the PC algorithm to learn a causal graph from service metrics. However, the resulting graph is often an equivalence class with undirected edges, prompting researchers to arbitrarily convert it into a directed acyclic graph (DAG). RUN [Lin et al., 2024], for example, removes the edge between two nodes with the lowest correlation, but this method does not guarantee the representation of the true underlying graph. In the second phase, existing algorithms *rank* all nodes using graph centrality measures such as random walk [Wang et al., 2018, Ma et al., 2020], PageRank [Wu et al., 2021, Xin et al., 2023, Lin et al., 2024], BFS [Lin et al., 2018], and DFS [Chen et al., 2014]. However, many rely on arbitrary objective functions that may not accurately reflect the failure propagation chain. For example, Groot [Wang et al., 2021] assumes that sink nodes are more likely to be root causes and assigns them different scores than others.

During normal operations, site engineers or RCA systems can proactively prepare for potential failures by learning cause-effect relationships through domain knowledge or causal discovery from observational data, a topic extensively explored in the literature [Spirtes and Glymour, 1991, Spirtes et al., 2000, Chickering, 2002, Peters et al., 2013, Zheng et al., 2018, Lam et al., 2022]. In this context, observational data refers to metrics collected before a failure occurs, while post-interventional data pertains to metrics gathered after the failure. Recent work by Budhathoki et al. [2022] leverages a graph from normal hours and allows anomalous samples from multiple distributions but assumes known, invertible functional relations, which are hard to estimate. Okati et al. [2024] extends this by relaxing the functional assumptions. Nonetheless, a fully known causal structure can be difficult to obtain in practice, especially in a large-scale system. Although Okati et al. [2024] has proposed a score function for RCA without any causal knowledge, it remains unclear how to incorporate existing causal discovery algorithms, which often give a partial causal structure.

Our contribution. In this paper, we introduce a novel algorithm, **Root Cause Analysis with Causal Graphs (RCG)**, which uses a system's normal operational period to proactively prepare for potential failures. To the best of our knowledge, this is the first work to show the advantages of using a *partial* causal structure learned from normal operation data for RCA. We achieve this by showing a reduced number and order of conditional independence (CI) tests required by our proposed algorithm against state-of-the-art RCA methods based on CI tests. We begin by examining the simplest case, where there is a single root cause and the causal relationships are fully known in the form of a DAG. Interestingly, we show that identifying the root cause in a DAG is equivalent to solving a well-established graph theory problem known as Interactive Graph Search (IGS) Tao et al. [2019], with minor modifications. This reduction to IGS provides a novel insight: a logarithmic number of marginal invariance tests relative to the number of variables is sufficient to identify root causes given a causal graph. For cases of multiple root causes, we propose another algorithm to learn the root cause of failure. Our algorithm exploits causal knowledge that is learned offline and can be represented as a mixed graph. For example, it can accept a CPDAG, a mixed graph with directed and undirected edges that represents a set of causal graphs containing the true graph. It can also accept partial causal structures obtained by testing a smaller set of conditional independences from data such as LOCI Wienöbst and Liskiewicz [2020], or the recently proposed kPC Kocaoglu [2023], which are shown to be effective in the data-scarce regime. We summarize our contributions below.

1. Considering a single root cause and given a complete causal structure of a system, we map the problem of RCA to IGS. We further provide an algorithm that identifies the root cause with $\mathcal{O}(\log_2(n) + d\log_{1+d} n)$ number of marginal invariance tests and show that any algorithm that solely relies on marginal invariance tests for RCA must perform $\Omega(\log_2(n) + d\log_{1+d} n)$ many tests, where n is the number of variables and d is the maximum degree in the graph.
2. In scenarios with multiple root causes, we propose an algorithm that leverages causal knowledge represented as a mixed graph (*e.g.*, CPDAG) learned *before* the failure. The algorithm efficiently finds the separating set based on the estimated structure along with an information-theoretic approach to identify the true root causes of failure. We also prove its soundness for RCA given a partial causal structure.
3. We validate the performance of our proposed algorithm by showing its higher accuracy relative to state-of-the-art methods, such as RCD [Ikram et al., 2022], RUN [Lin et al., 2024], and BARO [Pham et al., 2024], through experiments on a real-world production-level application, which has a large number of variables with limited failure samples.

2 BACKGROUND

In this section, we give the most relevant definitions. We use boldface letters to denote a set of random variables. For more details on other graph notations and terminology, please refer to Appendix A. We also discuss related work in Appendix B.

Causal Graphs. A *causal graph* is used to encapsulate the causal relationships among variables in the form of a DAG, where each node represents a variable X and the directed edge $X \rightarrow Y$ indicates that X causes Y. A variable is said to cause another variable if a change in the former induces a change in the probability distribution of the latter.

Structural Causal Models (SCMs) and Causal Bayesian Networks (CBNs). SCMs are used to model causality among a set of random variables. Each variable X is a function of some endogenous variables as its parents, denoted by $Pa(X)$, and an exogenous noise term, denoted as E_X e.g. $X = f_X(Pa(X), E_X)$. An SCM induces a causal graph by assigning a set of endogenous variables as the parents of X for all variables X. CBNs are used to define a causal model that specifies the observational and interventional distributions via the truncated factorization formula without the functional descriptions like SCMs in a causal graph.

D-separation, Markov Equivalence, CPDAG. In a causal graph D, a path p between X and Y is *d-connecting (active)* relative to a set of vertices $\mathbf{Z}(X, Y \notin \mathbf{Z})$ if (i) every non-collider on p is not in $\mathbf{Z}$ and (ii) every collider on p is an ancestor of some $Z \in \mathbf{Z}$. Otherwise, we say $\mathbf{Z}$ *blocks* p. If $\mathbf{Z}$ blocks all paths between X and Y, we say X and Y are *d-separated* relative to $\mathbf{Z}$, denoted by $(X \perp\!\!\!\perp Y|\mathbf{Z})_D$. Two DAGs are *Markov equivalent* if they share the same set of d-separation statements. The set of DAGs that are Markov equivalent is called a Markov equivalence class of DAGs, denoted by $[D]$. Generally, a DAG is only identifiable up to its Markov equivalence class since different DAGs can generate the same observational distribution. This leads to an important concept about a partial causal structure. A *completed partially directed acyclic graph* (CPDAG) that represents $[D]$ and has the same skeleton as D, with directed edges $X_i \to X_j$ if the edge direction between X_i and X_j holds for all DAGs in $[D]$, and undirected edges otherwise.

Possible Parents relative to Equivalence Class $[D]$. X is called a *possible parent* of Y, denoted as $PossPa_D(X)$, if any of the following edges is in D: $\{X - Y, X o\!\!\rightarrow Y, X \to Y, X o\!\!-\!\!o Y\}$. The notations $X o\!\!\rightarrow Y$ and $X o\!\!-\!\!o Y$ are only applicable for a particular partial structure known as k-essential graphs which we will discuss in greater details in Appendix D.

Intervention and F-NODE. An intervention on a variable is the process of changing the generative mechanism of that variable. Randomized controlled trials (RCTs) and A/B tests are the most common notion of interventions. Pearl uses do-operator $do(X = x)$ to capture this type of intervention. For instance, when $do(X = x)$ forces a variable X to take on certain values, it is known as the *hard* interventions [Pearl, 2009]. Its effect in a causal graph is to remove the edges incoming to the intervened nodes. It is different than another type of intervention known as the *soft interventions*, which do not completely alter the causal mechanisms and retain the original causal graph by only replacing $f_X(Pa(X), E_X)$ with $f'_X(Pa(X), E_X)$ where $f' \neq f$. A variable F-NODE has been extensively used to represent the effect of an intervention on a system [Pearl, 1995, Yang et al., 2018, Mooij et al., 2020]. Throughout this work, we denote a ground truth DAG D being augmented by F-NODE as an intervention to the root cause as D_{aug}. We will discuss its role in RCA in the next section. We assume no latent confounders. We also make the extended faithfulness assumption as in Jaber et al. [2020]. It means that any statistical independence implies d-separation. Please refer to Appendix A.6 and A.7 for more details.

3 PROBLEM FORMULATION

A system has n components $\mathcal{M} = \{m_1, \ldots, m_n\}$. Within a given time interval, the monitoring tool collects at least d metrics from each of the components, i.e. $\mathcal{T}(i, t) = \{r_{i,1,t}, \ldots, r_{i,d,t}\}$, where $d \geq 1; \forall i \in \{1, \ldots, n\}$, $\mathcal{T}(i, t)$ is a set of d metrics of component i at time instance t. Considering the entirety of the data, we have two time series datasets defined as $\mathcal{D} = \{\mathcal{T}(1, 1), \ldots, \mathcal{T}(n, t-1)\}$ and $\mathcal{D}^\star = \{\mathcal{T}(1, t), \ldots, \mathcal{T}(n, \gamma)\}$, where t represents the time when the failure was first registered and γ is the time when the issue was fixed. We consider the setting where one can learn some cause-effect relations in the form of a CPDAG at the time s from $\mathcal{D}$, where $s < t$. We leverage this partial causal structure to pinpoint the root cause between timestamps t and γ.

Failure as Interventions. An important observation of this problem is to model a failure as a soft intervention on the failing mode [Ikram et al., 2022]. Here, the representation of F-NODE allows one to identify the distribution invariances $P_N(X|Pa(X)) = P_A(X|Pa(X))$, where P_N and P_A are the distributions under normal mode of operation and anomalous operation respectively. By concatenating both of these datasets, one can sample from the distribution $P^\star$ of a set of observed variables $\mathbf{V}$ involving F-NODE, denoted as F, where $P^\star(\mathbf{V}|F = 0) = P_N(\mathbf{V})$ and $P^\star(\mathbf{V}|F = 1) = P_A(\mathbf{V})$. Under this formalism, the invariance $P_N(X|Pa(X)) = P_A(X|Pa(X))$ corresponds to conditional independence between X and F given $Pa(X)$. Since F-NODE cannot have any incoming edges, one can then employ a series of CI tests on the sampling distributions $\hat{P^\star}$ to determine which node is the root cause R (the child of F-NODE) by observing $(R \not\perp\!\!\!\perp F|Pa(R))_{\hat{P^\star}}$.

To demonstrate the advantages of leveraging normal operation time, we begin by examining the number and order of CI tests that can be reduced given a complete causal graph (*i.e.*, a DAG). Then, we will discuss a more practical solution for scenarios where only a partial causal structure is available during the failure period.

4 RCA WITH A KNOWN GRAPH

In this section, we highlight the advantages of leveraging a DAG prior to failure by contrasting it with RCD, an RCA method that relies solely on CI tests. We begin by addressing the primary limitation of RCD, specifically its inability to utilize a DAG, which leads to a significant increase in the number of CI tests required. Then, we introduce the use of graphical structures as a potential solution in the case of a single root cause. For details on RCD, see Appendix F. All proofs are provided in the Appendix C.

Firstly, RCD only learns the adjacencies between F-NODE and each observed variable as it operates. It conditions on

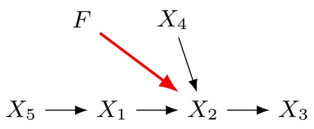

$$\begin{tikzcd}
 & F \arrow[dr, red] & X_4 \arrow[d] & \\
X_5 \arrow[r] & X_1 \arrow[r] & X_2 \arrow[r] & X_3
\end{tikzcd}$$

Figure 1: An example to show how Lemma 4.1 and 4.2 help identify the root cause with a few invariance tests given a causal graph, where X_1 is the root cause.

every possible subset $\mathbf{S}$ of variables $\mathbf{V}$ for testing the conditional independence relations between each pair of variables i.e., $X, Y \in \mathbf{V}$ until it identifies a conditioning set that yields conditional independence, thereby excluding a potential node as the root cause under Assumption A.7. However, under Assumption A.6, having access to a DAG D allows us to conduct n CI tests *e.g.*, $(F \perp\!\!\!\perp X|Pa_D(X))$ for each observed variable X where n is the number of observed variables. In other words, RCD performs at least as many CI tests as a naive approach using the DAG would require. Secondly, RCD may condition on a set of variables that is much larger than the actual parent set, resulting in unreliable CI test results in practice. In contrast, since our graphical structure captures ancestral relationships between nodes and there is only a single root cause variable, we argue that the root cause can be identified in fewer than n tests. To support this, we present key results that allow for a systematic exploration of the causal structure, significantly reducing the number of required CI tests.

For the case where there is only a single root cause, the following two lemmas indicate that certain CI relations can eliminate variables from being considered as root causes, under Assumption A.6 and A.7. The first lemma states that all ancestors of a variable X can be excluded as the root cause if we observe that F is conditionally independent of X given some variables $\mathbf{Z}$. The second lemma asserts that all non-ancestors of X can be excluded as the root cause if F is conditionally dependent on X. Unlike RCD, which performs a series of CI tests and stops once a CI relation excludes a variable as the root cause, our approach systematically eliminates variables using two key results—Lemma 4.1 and Lemma 4.2—without requiring tests on every variable. W

Lemma 4.1. *Given a DAG D, if $(F \perp\!\!\!\perp X)_P$ for some $X \in \mathbf{V}$, then $A \notin Ch_{D_{aug}}(F)$ for all $A \in An_D(X)$, where P is any joint distribution between variables on D_{aug}.*

Lemma 4.2. *Given a DAG D, if $(F \not\perp\!\!\!\perp X)_P$ for some $X \in \mathbf{V}$, then then $Q \notin Ch_{D_{aug}}(F)$ for all $Q \in NAn_D(X)$, where P is any joint distribution between variables on D_{aug}.*

We will use Figure 1 to illustrate how Lemmas 4.1 and 4.2 may help identify the root cause, which is X_1 in this case, with less than n invariance tests. We can start by arbitrarily picking a variable for testing conditional independence with F. Suppose we select X_2 to test whether $(F \perp\!\!\!\perp X_2)_P$. By Assumption A.7, we will observe $(F \not\perp\!\!\!\perp X_2)_P$. Then, Lemma 4.2 says that X_3 cannot be the root cause. Suppose we pick X_1 to test for conditional independence, then we will observe $(F \perp\!\!\!\perp X_1)_P$. Then, by Lemma 4.1, we know that X_5 cannot be the root cause either. Then, we are only left with X_4 to test for conditional independence. This results in a total of 3 marginal independence tests, which is less than $n = 5$. To further illustrate the utility of these two key results, we show that there is a one-to-one correspondence between the use of marginal invariance tests for RCA with a known DAG and the problem known as *Interactive Graph Search* (IGS) [Tao et al., 2019] which guarantees identification of the root cause in fewer than n tests. For the sake of clarity, we provide the formal problem formulation of IGS.

Interactive Graph Search (IGS)
INSTANCE: A DAG $D = (\mathbf{V}, \mathbf{E})$ that has a single root node, an adversary chooses arbitrarily a target node $R \in \mathbf{V}$. There is an oracle that returns a boolean answer to the given query: yes, if there is a directed path from X to R and no otherwise for any $X \in \mathbf{V}$.
QUESTION: *What is the minimum number of queries to ask in order to identify R in D?*

Lemma 4.3. *Consider a DAG $D = (\mathbf{V}, \mathbf{E})$ with a single sink node and D' be a DAG by reversing every edge direction in $\varepsilon_k()$, let $Q(X)$ be a query to the oracle on whether some $X \in \mathbf{V}$ has a directed path to an unknown target node $R \in \mathbf{V}$.*

$$Q(X) = \textit{yes} \Leftrightarrow (F \not\perp\!\!\!\perp X)_P \tag{1}$$

Therefore, if $Q(X) = $ yes, then $X \in An_{D'}(R)$. If $Q(X) = $ no, then $X \in NA_{D'}(R)$.

The significance of Lemma 4.3 is that a solution to IGS is now a solution to RCA using marginal invariance tests, given a known DAG. For DAGs that do not have a single sink node, we can simply add a dummy node as a child of all the sink nodes. Hence, the following theorem is an immediate consequence of Theorem 1 (see Appendix C.1) proven by Shangqi et al. [2023]).

Theorem 4.4. *Given a DAG D with a single sink node, any algorithm the only uses marginal invariance tests must perform $\Omega(\log_2 n + d\log_{1+d} n)$ many tests to find the single root cause in the worst case, where d is the maximum in-degree of D and n is the number of nodes. There exists an algorithm that finds the root cause with $\mathcal{O}(\log_2 n + d\log_{1+d} n)$ marginal invariance tests.*

Shangqi et al. [2023] provides an optimal algorithm that bounds the worst-case number of queries to $\mathcal{O}(\log_2 n + d\log_{1+d} n)$ for IGS. Due to Lemma 4.3, this algorithm can be modified for RCA with a single root cause using marginal

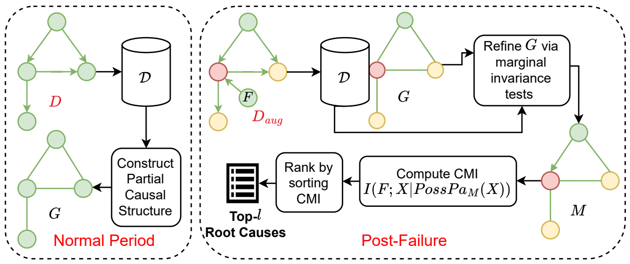


Figure 2: RCG framework: The true graphs, D and D_{aug}, are unknown to the algorithm. Red nodes represent the root cause, while orange nodes are impacted but not the root cause. During the normal period, RCG learns a CPDAG from a sound causal discovery algorithm. After a failure, it identifies the root cause by performing marginal invariance tests to further orient the edges and computing the Conditional Mutual Information (CMI), denoted by I, between the F and each node in the graph. Finally, RCG ranks the nodes by CMI scores, outputting an ordered list of potential root causes.

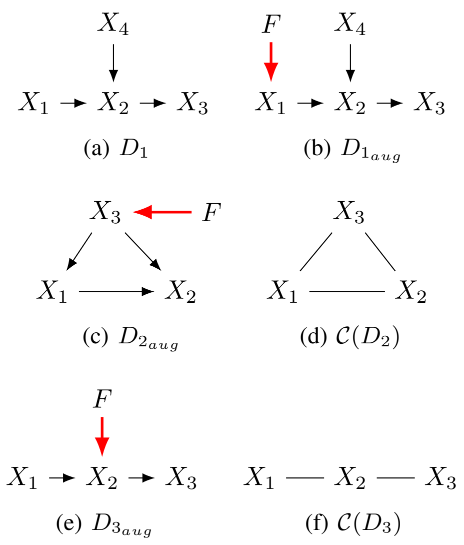


Figure 3: (a) A true graph D_1 which is also the CPDAG of D_1. (b) A true graph augmented from D_1. It shows how a CPDAG can help identify root causes more efficiently. (c)-(d) A true graph $D_{2_{aug}}$ augmented from D_2 and the CPDAG $\mathcal{C}(D_2)$. They show how a CPDAG may not help identify root causes with more CI tests since it does not have any orientations. (e)-(f) A true graph $D_{3_{aug}}$ augmented from D_3 and the CPDAG $\mathcal{C}(D_3)$, showing that not all CPDAGs without orientations are equally informative for RCA.

invariance tests. Hence, we have shown that we need fewer than n tests and that marginal invariance tests alone are sufficient for identifying the root cause given a DAG. We provide the pseudocode of the modified IGS algorithm through Algorithm 4 in the Appendix.

5 RCA WITH A PARTIAL GRAPH

Having established that a causal graph helps to reduce the number and order of CI tests, we now turn our attention to the challenge of performing RCA with partial graphical structure in the case of multiple root causes. We provide the workflow of the proposed solution in Figure 2. All proofs are provided in the Appendix C. We start with an example to highlight three main challenges of incorporating a partial causal structure with CI tests, as illustrated in Figure 3. For simplicity, we use a CPDAG as the given partial causal structure. However, our results in this section also hold for other partial causal structures, which we leave to Appendix D. We briefly discuss the challenges of learning CPDAG from observational data on top of RCA and the benefits of using other partial causal structures in the end of Appendix B.

Motivating Example. Consider the true augmented graph $D_{1_{aug}}$ shown in Figure 3(b). The CPDAG of D_1 is the induced subgraph of $D_{1_{aug}}$ obtained by removing F. Here, we can use a single CI test to identify the root cause. We can select X_1 and test the CI relation $(F \perp\!\!\!\perp X_1)_P$. Since X_2, X_3, X_4 are non-ancestors of X_1 in the CPDAG and $(F \not\perp\!\!\!\perp X_1)_P$, it follows that X_1 must be a child of F in the ground truth. Hence, X_1 is the root cause. However, RCD must have tested 6 CI tests e.g., $(F \perp\!\!\!\perp X_4)_P$, $(F \not\perp\!\!\!\perp X_2)_P$, $(F \perp\!\!\!\perp X_2|X_1)_P$, $(F \not\perp\!\!\!\perp X_3)_P$ and $(F \perp\!\!\!\perp X_3|X_2)_P$ (or $(F \perp\!\!\!\perp X_3|X_1)_P$), in order to conclude that X_1 is the root cause in the best case. Nonetheless, it is unclear how to initially select a variable for testing conditional independence. The second challenge is that some CPDAGs do not have any orientations as shown by Figure 3(d). We cannot utilize any ancestral relationships even if we exhaust all marginal tests. The third challenge is that all CPDAGs that do not have any orientations are not equally informative for RCA. Consider another true augmented graph in Figure 3(e) and the corresponding CPDAG learned from observed data in 3(f), one can infer that: (i) F cannot point to X_1 due to $(F \perp\!\!\!\perp X_1)_P$; (ii) F has a directed path to X_2. Therefore, $X_1 - X_2$ can be further oriented as $X_1 \to X_2$ in Figure 3(f) with interventional data. Since all the unshielded colliders in Figure 3(f) should have been oriented, $X_2 - X_3$ can then be further oriented as $X_2 \to X_3$, resulting in $X_1 \to X_2 \to X_3$. Hence, we can conclude that X_2 is the root cause as X_2 is the parent of X_3 without testing whether $(F \perp\!\!\!\perp X_3)_P$ holds.

Ranking Root Causes. A key requirement for RCA tools is the output format. While failures typically have few root causes, much of the literature focuses on ranking nodes and reporting the top-l. This poses a challenge for approaches that rely on CI tests, which often identify only a single or a few root cause nodes. RCD addresses this by gradually increasing the significance level, α, in its CI tests and rerunning the algorithm until at least l nodes are identified.

However, this does not guarantee a meaningful ranking; the resulting nodes may appear in an arbitrary order, and multiple reruns increase runtime.

To address this along with the challenges mentioned previously , RCG (Algorithm 1) leverages the critical insight that the ranking in RCA aligns with an information-theoretic approach. Clearly, any non-root-cause variable $\bar{R}$ can be d-separated from F given its parents $Pa_{\bar{R}}$, while only the true root cause R is d-connecting with F given its parents $Pa(R)$. Under the faithfulness assumption, F must be conditionally dependent with R given $Pa(R)$, and by Assumption A.6, F must be conditionally independent with $\bar{R}$ given $Pa(\bar{R})$. These conditional independencies can be measured using CMI. Thus, RCA with a partial causal structure can be broken down into two steps: finding the parents of each variable and estimating the CMI given its parents. Ranking the potential root causes is achieved by sorting the CMI values in descending order. This non-parametric method is robust, capturing both linear and nonlinear dependencies, and works across various types of distributions, whether discrete, continuous, or mixture.

However, given a CPDAG, the parent set of each variable may not always be known. Our key contribution is to show that computing the CMI between F and each variable X, conditioned on possible parent set of X, is sufficient to identify root causes. This allows us to identify the root cause using only n invariance tests. We have proven the soundness of our algorithm for identifying root causes. This result is further extended to other partial causal structures in the form of a mixed graph such as k-CPDAG Wienöbst and Liskiewicz [2020] or k-essential graphs Kocaoglu [2023] for the data-scarce regime, which we discuss in Appendix D. Our algorithm can accept the output of *any* causal discovery algorithm, once it is converted to a CPDAG [1]. To combat finite sample noise, Algorithm 1 first sorts the mutual information between F and each variable in descending order. Then, it starts by using the minimum mutual information as a threshold to determine statistical independence and orient the CPDAG (lines 5-12). It repeatedly increments the threshold based on the next smallest mutual information until we have a consistent ranking of the root causes (see lines 19-20), meaning that there cannot be any variable that has a low mutual information with F but a high CMI given its possible parents. We do so to ensure the orientation applied to the CPDAG is consistent with the ranking procedure.

Theorem 5.1. *Given a CPDAG output by any sound causal discovery algorithms and under causal sufficiency and the extended faithfulness assumption, Algorithm 1 returns the true root cause variables.*

[1]A CPDAG captures all edges that can be learned through CI constraints and the remaining edges are uninformative.

Algorithm 1 RCA with Causal Graphs (RCG)

```
input  Observational data $\mathcal{D}$, interventional data $\mathcal{D}^\star$, a CPDAG
       $\mathcal{C}(D) = (\mathbf{V}, \mathbf{E})$, Max. no of root causes $l$,
output Top-$l$ root causes
 1: Concatenate $\mathcal{D}$ and $\mathcal{D}^\star$ with a binary indicator variable $F$.
 2: for $X \in \mathbf{V}$ do
 3:     $A_X \leftarrow I(F;X)$
 4: $A \leftarrow$Sort $X \in \mathbf{V}$ by $A_X$ in ascending order
 5: Create an empty list $\mathbf{V}_s^\star$
 6: for $\alpha \in A$ do
 7:     $G \leftarrow \mathcal{C}(D)$
 8:     for $X, Y \in \mathbf{V}$ do
 9:         if $I(F;X) < \alpha$ and $I(F;Y) \geq \alpha$ then
10:             If $X \leftarrow Y$ is in $G$, remove $X \leftarrow Y$
11:             If $X - Y$ is in $G$, orient $X \rightarrow Y$
12:     for $X \in \mathbf{V}$ do
13:         $CMI_X \leftarrow I(F; X|PossPa_G(X))$
14:     $\mathbf{V}_s \leftarrow$Sort $X \in \mathbf{V}$ by $CMI_X$ in descending order
15:     if $\exists X$ that has $I(F;X) < \alpha$ and $CMI_X$ is ranked on
        top-$l$ in $\mathbf{V}_s$ then
16:         Return the first $l$ root causes from $\mathbf{V}_s^\star$.
17:     $\mathbf{V}_s^\star \leftarrow \mathbf{V}_s$
18: Return the first $l$ root causes from $\mathbf{V}_s^\star$.
```

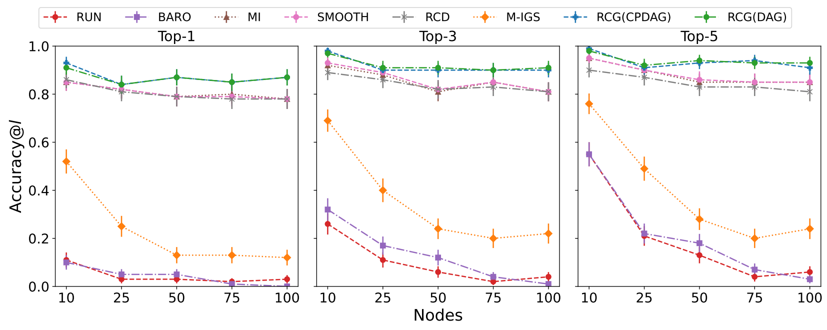
6 EXPERIMENTS

In this section, we evaluate RCG by addressing two key questions: 1) *Does a causal graph help RCG identify the root cause?* 2) *How quickly can RCG find the root cause?* We then discuss our implementation setup and present the results. We provide additional results in Appendix G.

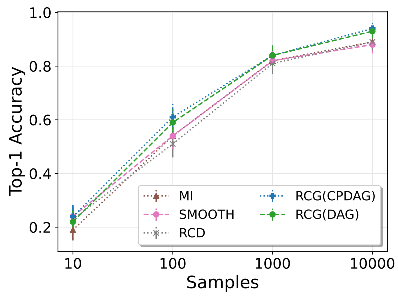
Implementation. To generate experimental data, we followed a streamlined approach [Ikram et al., 2022, Lin et al., 2024], using `pyagrum` [Ducamp et al., 2020] to create random causal graphs. We then generated samples for both observational and interventional settings by perturbing the data generation process of a randomly selected node. To ensure robustness, each experiment was repeated 100 times, with results reported as the mean and standard error. In RCA literature, a key metric for evaluating effectiveness is accuracy at top-l, defined as the probability of identifying the root cause within the top l ranked causes. Hence, we report top-l accuracy along with the execution runtime.

We implemented the following baselines:

- **RUN** [Lin et al., 2024]: It constructs a causal graph using neural Granger causal discovery with contrastive learning. It ranks the nodes by PageRank with a personalized vector according to the learned graph.
- **BARO** [Pham et al., 2024]: A non-causal approach that ranks root causes by computing a score for each variable, based on the absolute difference between each post-failure sample and the median of pre-failure data.
- **SMOOTH** Okati et al. [2024]: A recent work that tries to find the root cause given a complete causal graph.



(a) Top-l accuracy of RCG compared to baselines.



(b) Top-1 accuracy with varying samples

Figure 4: The results demonstrate that RCG with RCG-2 consistently achieves higher accuracy than RCD. While MI struggles due to its inability to condition on the parents of each node, whereas RCD can condition on other nodes but lacks information about the causal structure. In contrast, RCG overcomes these challenges by learning a causal graph and using CMI to rank the nodes effectively.

- **MI**: A simple approach that sorts each node based on its mutual information with F.
- **RCD** [Ikram et al., 2022]: A recent method that uses CI tests to identify the root cause.
- **RCG**: A prototype of Algorithm 1 that uses a CPDAG.

To demonstrate the value of graphical structure, we first present an experiment where all baselines used the ground truth graph as input. The results with graphs learned from data are shown in Appendix G.2. We also compare two variants of RCG: M-IGS (Modified IGS)[2], which takes a DAG as input and identifies the root cause per Lemma 4.3; and RCG(CPDAG), which uses the essential graph generated by the PC algorithm [Spirtes et al., 2000]. Furthermore, we used 10,000 samples for the normal period and only 1,000 samples for the post-failure dataset.

Figure 4(a) shows the top-l accuracy of different approaches with $l = 1/3/5$. Although M-IGS offers the lowest runtime among all CI-based methods, its accuracy drops sharply. This decline stems from a key limitation: *M-IGS assumes perfect CI tests*, but in practice, test results can be unreliable due to limited sample availability. As a result, M-IGS often makes incorrect decisions, leading to poor performance, especially as the number of nodes increases. This weakness is particularly evident when comparing M-IGS with SMOOTH. Both methods operate on a fully known causal graph (DAG), but SMOOTH, being score-based, ranks variables using the anomaly scores of their parents and is more robust to noisy CI tests. Consequently, it outperforms M-IGS under imperfect conditions. This observation underscores a critical point: while IGS has strong theoretical guarantees, its practical performance suffers in the presence of noisy CI tests, where even a single error can propagate and significantly impact results.

Similarly, RUN performs poorly due to its PageRank personalization algorithm, which incorporates arbitrary constraints not applicable to our experimental setup, such as assuming that leaf nodes are more likely to be the root cause. As a result, even with the ground truth DAG, RUN fails to identify the root cause. Also, BARO, which relies solely on the absolute difference between normal and failure periods, performs poorly because failures can propagate across multiple nodes in a graph, often having a more pronounced impact on child nodes than on the root cause itself. When comparing RCD and RCG, we find that RCG generally achieves better accuracy. With 100 nodes, RCG(CPDAG) identifies the root cause in the top-1 position with an accuracy of 87%, surpassing RCD's 78%. This can further improve to 93% when ranking top-5 nodes. Similarly, when comparing RCG (CPDAG) with SMOOTH, we observe that RCG consistently outperforms SMOOTH. For instance, at 100 nodes, RCG achieves an accuracy of 87%, compared to 78% for SMOOTH in finding root cause in top-1. A key distinction between the two methods is that RCG(CPDAG) operates on a partially known causal graph, whereas SMOOTH requires a fully specified DAG as input.

Since root cause analysis is often time-sensitive, we report the number of samples required for each approach to perform effectively. Instead of execution time, we focus on sample efficiency, as most graph-based methods can be parallelized in large clusters. Hence, the key question is how many samples need to be collected before an approach becomes effective. Nevertheless, we provide runtime comparisons in Appendix G.4. Figure 4(b) shows the top-1 accuracy of three competing approaches on 25-node graphs with a varying number of interventional samples.

[2]For IGS, we referenced the recent findings from the POMS paper [Shangqi et al., 2023], but the authors declined to share their code in a way that can be made public. Consequently, we implemented an older, simpler version from [Tao et al., 2019]. For a runtime comparison, please see Theorem C.1 and C.2 in Appendix.

RCG(CPDAG) consistently outperforms baselines due to its reliance on the CPDAG and its ability to orient edges after failure. Appendix G.3 extends these results to 50- and 100-node graphs.

7 CASE STUDY

Sock-shop. This section demonstrates the effectiveness of RCG using the Sock-shop application, a microservice-based replica of a web application for selling socks. The system consists of 13 microservices, with 5 being the most critical and user-facing. Although Sock-shop is microservice-based, our method remains system-agnostic. We used the dataset from Ikram et al. [2022], which includes two failure types: CPU hog and memory leak. The dataset contains 50 instances, each running for 5 minutes in both normal and failure conditions. We repeated each experiment 50 times and report the mean top-l accuracy.

For Sock-shop scenario, we considered state-of-the-art RCA baselines, including causalRCA [Xin et al., 2023] (shown as cRCA), RUN [Lin et al., 2024], BARO [Pham et al., 2024], and RCD. To demonstrate the utility of the extended version of Algorithm 1 (see Algorithm 2 in Appendix E), we used a recent causal discovery algorithm k-PC Kocaoglu [2023], which learns the graph using a restricted conditioning set. Note that our algorithm only requires a partial causal graph, and k-PC was only chosen for its ability to constrain the separating set size, helping to prevent faithfulness violations in real data. We use postfixes to indicate how k was chosen. For example, RCG-k refers to using all conditioning sets up to size k, where $k \in \{0, 1\}$. Additionally, we construct a causal graph based on the system's call graph and use that as the input to Algorithm 1. We denote this algorithm by RCG-Expert, which uses expert knowledge for identifying the root cause. In this case, the possible parent set in Algorithm 1 is the parent set indicated by the DAG structure.

Table 1 shows the top-l accuracy of RCG with different baselines for CPU hog failure. We report the results for the memory leak failure in the Appendix H. The results align with those from our synthetic data experiments in Section 6. Notably, RCD and RCG-0 perform similarly because, with $k = 0$, k-PC is limited to marginal CI tests, producing a dense k-essential graph (see Appendix D. This leads to a larger set of possible parent nodes, forcing RCG to condition on more variables, which can obscure the true root cause. However, when $k = 1$, RCG outperforms RCD as k-PC is allowed to use separating sets of size up to one. Increasing k improves graph learning but demands larger sample sizes for reliable CI tests. Additionally, the system call graph shows that RCG achieves high accuracy when a high-quality causal graph is learned from observational data.

Notably, the performance of BARO, which achieves high top-1 accuracy on the Sock-shop dataset for both failure types. However, this contradicts our earlier experiments in Section 6. As we show in the next section with real-world data, BARO performs poorly. This discrepancy suggests that BARO may be overfitting to the Sock-shop dataset.

Real Datasets. We collected data from a real-world production application from January to July 2024, during which four outages were reported. For each incident, Software Reliability Engineers (SREs) documented key details, including outage duration, detection time, resolution method, and root cause (see Table 3). We presented the SREs with the top 10 ranked nodes from each baseline and asked them to confirm if the true root cause was among them. We report the rank of the root cause for each incident, where a lower rank indicates better performance by the method.

Table 2 compares the performance of MI, BARO, RCD, and RCG on the real-world dataset. The results show that RCG-0 consistently outperforms BARO and matches closely with MI's performance in outages C and D. MI's strong performance may stem from data inequality processing, but it breaks down when confounders ext between root causes and their descendants. In outage B, RCG-0 ranks the root cause first, while MI places it ninth. For outage A, RCG-0 identifies the root cause within the top 10, whereas MI does not. BARO often ranked the root cause near the bottom and failed to identify it entirely in one case, highlighting the limitations of methods focused solely on detecting noticeable changes. This also underscores the drawbacks of relying on a single point of the distribution (such as the median), which may not accurately capture the shift between the two distributions. RCD performs well on outages C and D, but it took approximately 25 minutes to produce results, whereas RCG completed in 10 seconds. We also find that increasing k did not consistently improve accuracy by comparing RCG-0 with RCG-1 and RCG-2. In some cases, accuracy declined due to less reliable CI tests with larger separating sets, leading to incorrect parent node conditioning and inaccurate rankings. This aligns with our insight that relying on high-order CI tests with small interventional samples is not ideal. In the additional experiment in Appendix G.1, we demonstrate that RCD heavily relies on high-order CI tests after failure, which compromises its performance. In contrast, our extended version of RCG (see Algorithm 2) leverages a partial causal structure learned from observed data using CI tests with small conditioning sets and requires only marginal invariance tests after failure.

For RCG, we note that there is a trade-off between the time complexity of computing CMI and the granularity of the input structure provided to RCG if we take the time to obtain the input into account. For example, when using the PC algorithm Spirtes et al. [2000] to construct a CPDAG as input, the resulting graph is typically sparser than a k-essential graph, a graphical representation that only uses CI tests up to order k to construct, leading to more efficient CMI computations due to smaller conditioning sets. In con-

		MI	cRCA	RUN	BARO	RCD	RCG-0	RCG-1	RCG-Expert
top-1	Carts	0.79	0.80	0.00	1.00	0.56	0.40	0.81	0.30
	Catalogue	0.11	0.40	0.00	1.00	0.18	0.98	0.09	0.81
	Orders	0.36	0.40	0.00	1.00	0.68	0.57	0.37	0.96
	Payment	0.27	0.40	0.00	1.00	0.65	0.76	0.24	0.93
	User	1.00	1.00	0.00	1.00	1.00	0.56	1.00	0.87
	Avg.	0.51	0.60	0.00	1.00	0.61	0.66	0.50	0.77
top-3	Carts	1.00	0.80	0.42	1.00	0.87	0.42	1.00	1.00
	Catalogue	0.92	0.60	0.39	1.00	0.47	1.00	0.45	1.00
	Orders	1.00	0.40	0.07	1.00	0.92	0.6	1.00	1.00
	Payment	1.00	0.40	0.14	1.00	0.88	0.76	1.00	1.00
	User	1.00	1.00	0.06	1.00	1.00	0.64	1.00	1.00
	Avg.	0.98	0.64	0.22	1.00	0.77	0.68	0.89	1.00
top-5	Carts	1.00	1.00	0.62	1.00	0.82	0.42	1.00	1.00
	Catalogue	0.93	0.60	0.58	1.00	0.51	1.00	0.58	1.00
	Orders	1.00	0.60	0.15	1.00	0.87	0.62	1.00	1.00
	Payment	1.00	0.40	0.20	1.00	0.86	0.76	1.00	1.00
	User	1.00	1.00	0.11	1.00	1.00	0.70	1.00	1.00
	Avg.	0.99	0.72	0.33	1.00	0.81	0.70	0.92	1.00

Table 1: The table shows the top-l accuracy of different baselines on the data collected from the Sock-shop application after injecting a CPU hog into a given microservice.

Outage	RCG-0	RCG-1	RCG-2	MI	BARO	RCD
A	**7**	-	-	-	9	-
B	**1**	6	-	9	6	-
C	**1**	**1**	**1**	**1**	8	**1**
D	5	5	6	3	-	**2**

Table 2: Rank of the root cause among the top 10 nodes for each baseline, with a rank of 1 indicating the highest-ranked node and a dash indicating the root cause was not found. RCG-0 consistently outperforms BARO and performs comparably with MI, but higher values of k lead to a less reliable causal graph and decreased consistency.

Outage	Nodes	Normal Samples	Failure Samples	Duration (hours)
A	152	4783	918	15
B	141	4626	1217	20
C	149	3464	110	2
D	146	7165	567	5

Table 3: Summary of outages from a real-world production application.

trast, a k-essential graph often results in larger conditioning sets, making CMI computations more costly. This efficiency gap, however, is not obvious when CPDAGs are obtained from other discovery algorithms besides PC.

8 CONCLUSION

Identifying the root cause of system failures is a critical challenge in software systems. We argue that leveraging a system's partial causal structure can provide valuable insights for diagnosing failures. First, we demonstrate the value of the causal graph by showing that it can significantly reduce the number of invariance tests required. We establish a lower bound on the number of marginal CI tests necessary to identify the root cause, given the correct causal graph, for any algorithm that relies solely on marginal invariance tests. Then, we argue that a system's normal operational time can be leveraged to learn a partial causal graph. Based on this, we introduce an algorithm that systematically utilizes the partial causal graph to identify the root cause using a linear number of invariance tests. Our proposed algorithm RCG can incorporate a wide range of causal discovery algorithms and its performance will improve alongside with the advancement of causal discovery from observational data. Empirical results show that our approach outperforms state-of-the-art methods, improving fault detectability. We believe RCG shows promise for RCA under latent confounding by integrating causal discovery algorithms capable of learning from observational data with unobserved confounders. An interesting future direction is to explore how to effectively leverage multiple unknown interventional distributions for RCA within the framework of modeling the failure as a soft intervention. As mutual information performs well in our experiments, it is also worthwhile to explore the conditions under which mutual information will be most efficient in terms of both sample and time complexity for RCA.

Acknowledgements

This research has been supported in part by NSF CAREER 2239375, IIS 2348717, Amazon Research Award and Adobe Research. This material is based in part upon work supported by the National Science Foundation under Grant Number CNS-2333487, Army Research Lab under Contract number W911NF-20-2-0026, and gift funding from Adobe Research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

Bryan Andrews, Joseph Ramsey, Ruben Sanchez Romero, Jazmin Camchong, and Erich Kummerfeld. Fast scalable and accurate discovery of dags using the best order score search and grow shrink trees. *Advances in Neural Information Processing Systems*, 36:63945–63956, 2023.

Charles K Assaad, Imad Ez-Zejjari, and Lei Zan. Root cause identification for collective anomalies in time series given an acyclic summary causal graph with loops. In *International Conference on Artificial Intelligence and Statistics*, pages 8395–8404. PMLR, 2023.

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

Kailash Budhathoki, Lenon Minorics, Patrick Blöbaum, and Dominik Janzing. Causal structure-based root cause analysis of outliers. In *International Conference on Machine Learning*, pages 2357–2369. PMLR, 2022.

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

Sarthak Chakraborty, Shaddy Garg, Shubham Agarwal, Ayush Chauhan, and Shiv Kumar Saini. Causil: Causal graph for instance level microservice data. In *Proceedings of the ACM Web Conference 2023*, pages 2905–2915, 2023.

Pengfei Chen, Yong Qi, Pengfei Zheng, and Di Hou. Causeinfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 1887–1895. IEEE, 2014.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

Max Chickering. Statistically efficient greedy equivalence search. In *Conference on Uncertainty in Artificial Intelligence*, pages 241–249. Pmlr, 2020.

Max Chickering, David Heckerman, and Chris Meek. Large-sample learning of bayesian networks is np-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4027–4035, 2021.

Martin Drasar and Tomas Jirsik. It operations analytics: Root cause analysis via complex event processing. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 741–742. IEEE, 2019.

Gaspard Ducamp, Christophe Gonzales, and Pierre-Henri Wuillemin. aGrUM/pyAgrum : a Toolbox to Build Models and Algorithms for Probabilistic Graphical Models in Python. In *10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 609–612, Skørping, Denmark, September 2020. URL `https://hal.archives-ouvertes.fr/hal-03135721`.

Scott Emmons, Coburn Watson, and Brendan Gregg. A microscope on microservices. `netflixtechblog.com/a-microscope-on-microservices-923b906103f4/`, 2022. Netflix Technology Blog.

Yu Gan, Mingyu Liang, Sundar Dev, David Lo, and Christina Delimitrou. Sage: practical and scalable ml-driven performance debugging in microservices. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 135–151, 2021.

Chang Gong, Di Yao, Jin Wang, Wenbin Li, Lanting Fang, Yongtao Xie, Kaiyu Feng, Peng Han, and Jingping Bi. Porca: Root cause analysis with partially. *arXiv preprint arXiv:2407.05869*, 2024.

Azam Ikram, Sarthak Chakraborty, Subrata Mitra, Shiv Saini, Saurabh Bagchi, and Murat Kocaoglu. Root cause analysis of failures in microservices through causal discovery. *Advances in Neural Information Processing Systems*, 35:31158–31170, 2022.

Amin Jaber, Murat Kocaoglu, Karthikeyan Shanmugam, and Elias Bareinboim. Causal discovery from soft interventions with unknown targets: Characterization and learning. *Advances in neural information processing systems*, 33: 9551–9561, 2020.

Kathryn M Kellogg, Zach Hettinger, Manish Shah, Robert L Wears, Craig R Sellers, Melissa Squires, and Rollin J Fairbanks. Our current approach to root cause analysis: is it contributing to our failure to improve patient safety? *BMJ quality & safety*, 2016.

Murat Kocaoglu. Characterization and learning of causal graphs with small conditioning sets. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, pages 1–12, 2023.

Wai-Yin Lam, Bryan Andrews, and Joseph Ramsey. Greedy relaxations of the sparsest permutation algorithm. In *Uncertainty in Artificial Intelligence*, pages 1052–1062. PMLR, 2022.

Robert J Latino. How is the effectiveness of root cause analysis measured in healthcare? *Journal of Healthcare Risk Management*, 35(2):21–30, 2015.

Kenneth Lee, Bruno Ribeiro, and Murat Kocaoglu. Constraint-based causal discovery from a collection of conditioning sets. In *9th Causal Inference Workshop at UAI 2024*, 2024.

Mingjie Li, Zeyan Li, Kanglin Yin, Xiaohui Nie, Wenchi Zhang, Kaixin Sui, and Dan Pei. 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*, pages 3230–3240, 2022.

Cheng-Ming Lin, Ching Chang, Wei-Yao Wang, Kuang-Da Wang, and Wen-Chih Peng. Root cause analysis in microservice using neural granger causal discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 206–213, 2024.

JinJin Lin, Pengfei Chen, and Zibin Zheng. Microscope: Pinpoint performance issues with causal graphs in microservice environments. In *Service-Oriented Computing: 16th International Conference, ICSOC 2018, Hangzhou, China, November 12-15, 2018, Proceedings 16*, pages 3–20. Springer, 2018.

Meng Ma, Jingmin Xu, Yuan Wang, Pengfei Chen, Zonghua Zhang, and Ping Wang. Automap: Diagnose your microservice-based web applications automatically. In *Proceedings of The Web Conference 2020*, pages 246–258, 2020.

Yuan Meng, Shenglin Zhang, Yongqian Sun, Ruru Zhang, Zhilong Hu, Yiyin Zhang, Chenyang Jia, Zhaogang Wang, and Dan Pei. Localizing failure root causes in a microservice through causality inference. In *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2020.

Francesco Montagna, Nicoletta Noceti, Lorenzo Rosasco, Kun Zhang, and Francesco Locatello. Scalable causal discovery with score matching. In *Conference on Causal Learning and Reasoning*, pages 752–771. PMLR, 2023.

Joris M Mooij, Sara Magliacane, and Tom Claassen. Joint causal inference from multiple contexts. *Journal of Machine Learning Research*, 21(99):1–108, 2020.

Achille Nazaret and David Blei. Extremely greedy equivalence search. In *Uncertainty in Artificial Intelligence*, 2024.

Nastaran Okati, Sergio Hernan Garrido Mejia, William Roy Orchard, Patrick Blöbaum, and Dominik Janzing. Root cause analysis of outliers with missing structural knowledge. *arXiv e-prints*, pages arXiv–2406, 2024.

Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

Judea Pearl. Causal inference in statistics: An overview. 2009.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Causal inference on time series using restricted structural equation models. *Advances in neural information processing systems*, 26, 2013.

Luan Pham, Huong Ha, and Hongyu Zhang. Baro: Robust root cause analysis for microservices via multivariate bayesian online change point detection. *Proceedings of the ACM on Software Engineering*, 1(FSE):2214–2237, 2024.

Juan Qiu, Qingfeng Du, Kanglin Yin, Shuang-Li Zhang, and Chongshu Qian. A causality mining and knowledge graph based method of root cause diagnosis for performance anomaly in cloud applications. *Applied Sciences*, 10(6): 2166, 2020.

Joseph Ramsey, Madelyn Glymour, Ruben Sanchez-Romero, and Clark Glymour. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International journal of data science and analytics*, 3: 121–129, 2017.

Marc Schaaf, Gwendolin Wilke, Topi Mikkola, Erik Bunn, Ilkka Hela, Holger Wache, and Stella Gatziu Grivas. Towards a timely root cause analysis for complex situations in large scale telecommunications networks. *Procedia Computer Science*, 60:160–169, 2015.

Rajen D Shah and Jonas Peters. The hardness of conditional independence testing and the generalised covariance measure. 2020.

Lu Shangqi, Wim Martens, Matthias Niewerth, and Yufei Tao. Partial order multiway search. *ACM Transactions on Database Systems*, 48(4):1–31, 2023.

Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, Antti Kerminen, and Michael Jordan. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(10), 2006.

Peter Spirtes and Clark Glymour. An algorithm for fast recovery of sparse causal graphs. *Social science computer review*, 9(1):68, 1991.

Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search.* MIT press, 2000.

Eric V Strobl. Counterfactual formulation of patient-specific root causes of disease. *Journal of Biomedical Informatics*, 150:104585, 2024.

Eric V Strobl and Thomas A Lasko. Identifying patient-specific root causes with the heteroscedastic noise model. *Journal of Computational Science*, 72:102099, 2023.

Yufei Tao, Yuanbing Li, and Guoliang Li. Interactive graph search. In *Proceedings of the 2019 International Conference on Management of Data*, pages 1393–1410, 2019.

Dongjie Wang, Zhengzhang Chen, Yanjie Fu, Yanchi Liu, and Haifeng Chen. Incremental causal graph learning for online root cause analysis. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2269–2278, 2023a.

Dongjie Wang, Zhengzhang Chen, Jingchao Ni, Liang Tong, Zheng Wang, Yanjie Fu, and Haifeng Chen. Interdependent causal networks for root cause localization. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5051–5060, 2023b.

Hanzhang Wang, Zhengkai Wu, Huai Jiang, Yichao Huang, Jiamu Wang, Selcuk Kopru, and Tao Xie. Groot: An event-graph-based approach for root cause analysis in industrial settings. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 419–429. IEEE, 2021.

Ping Wang, Jingmin Xu, Meng Ma, Weilan Lin, Disheng Pan, Yuan Wang, and Pengfei Chen. Cloudranger: Root cause identification for cloud native systems. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 492–502. IEEE, 2018.

Kaitlynn M Whitney and Charles B Daniels. The root cause of failure in complex it projects: Complexity itself. *Procedia Computer Science*, 20:325–330, 2013.

Marcel Wienöbst and Maciej Liskiewicz. Recovering causal structures from low-order conditional independencies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10302–10309, 2020.

Li Wu, Johan Tordsson, Jasmin Bogatinovski, Erik Elmroth, and Odej Kao. Microdiag: Fine-grained performance diagnosis for microservice systems. In *2021 IEEE/ACM International Workshop on Cloud Intelligence (CloudIntelligence)*, pages 31–36. IEEE, 2021.

Ruyue Xin, Peng Chen, and Zhiming Zhao. Causalrca: causal inference based precise fine-grained root cause localization for microservice applications. *Journal of Systems and Software*, 203:111724, 2023.

Karren Yang, Abigail Katcoff, and Caroline Uhler. Characterizing and learning equivalence classes of causal dags under interventions. In *International Conference on Machine Learning*, pages 5541–5550. PMLR, 2018.

Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. *Advances in neural information processing systems*, 31, 2018.

Root Cause Analysis of Failure with Observational Causal Discovery (Supplementary Material)

Azam Ikram[†1] **Kenneth Lee**[†1] **Shubham Agarwal**[2] **Shiv Kumar Saini**[2] **Saurabh Bagchi**[1] **Murat Kocaoglu**[1]

[1]Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA
[2]Adobe Research, India

A GRAPH NOTATIONS

Definition A.1. A graph $D = (\mathbf{V}, \mathbf{E})$ consists of a set of nodes (variables) $\mathbf{V}$ and a set of edges $\mathbf{E}$. We use (X, Y) to denote an edge between a variable X and another variable Y in D. A directed graph has only directed edges ($\rightarrow$). A partially directed graph may have both undirected ($-$) and directed edges ($\rightarrow$). A graph $D' = (\mathbf{V}', \mathbf{E}')$ is a *subgraph* of $D = (\mathbf{V}, \mathbf{E})$ and D is a *supergraph* of D' if $\mathbf{V}' \subseteq \mathbf{V}$ and $\mathbf{E}' \subseteq \mathbf{E}$. D' is an *induced subgraph* of D if $\mathbf{E}'$ are all edges in $\mathbf{E}$ between nodes in $\mathbf{V}'$. A mixed graph consists of directed edges ($\rightarrow$), undirected edges ($-$) and bidirected edges ($\leftrightarrow$).

Definition A.2 (Path)**.** Two vertices in a graph are said to be *adjacent* if there is an edge between them. Given a partially directed graph D, a *path* from V_0 to V_n in D is a sequence of distinct vertices $\langle V_0, V_1, \dots, V_n \rangle$ such that for $0 \le i \le n-1$, V_i and V_{i+1} are adjacent. It is called a *causal* (or *directed*) path from V_0 to V_n in D if V_i is a parent of V_{i+1} for $0 \le i \le n-1$.

Definition A.3 (Skeleton)**.** A *skeleton* of a causal graph is the undirected graph obtained by making every adjacent pair connected via an undirected edge.

Definition A.4 (Colliders)**.** A consecutive triple of nodes $\langle X, Y, Z \rangle$ on a path is called a *collider* if both the edge between X and Y and the edge between Y and Z have arrowheads pointing to Y. If additionally X and Z are not adjacent, it is called *unshielded collider*. Any other consecutive triple is called a *non-collider*. If additionally, the two end vertices of the triple are not adjacent, it is called a *unshielded non-collider*.

Definition A.5 (Ancestrality)**.** In a graph D, for any two nodes X, Y in D, if there is a directed edge $X \rightarrow Y$, then X is a *parent* of Y and Y is a *child* of X in D. If there is a causal path from X to Y, then X is called an *ancestor* of Y and Y is called a *descendant* of X. We denote a set of parents of X, a set of children of X, a set of ancestors of X, a set of descendants of X and a set of non-descendants of X in D as $Pa_D(X)$, $Ch_D(X)$, $An_D(X)$, $De_D(X)$ and $NDe_D(X)$ respectively. By convention, X is both an ancestor and a descendant of X in D. A source (or root) node has no parents. A sink node does not have any child.

For learning D_{aug}, we need to leverage distributional invariances across the normal and anomalous datasets via the following two assumptions. For a more detailed discussion on these assumptions, please see Jaber et al. [2020].

Assumption A.6 (Ψ-Markov conditions)**.** Let $\mathbf{P}$ denote an ordered tuple of distributions and let $\mathcal{I}$ be an ordered tuple of the children of F-NODE. $\mathbf{P}$ is called Ψ*-Markov* relative to a graph $D_{aug} = (\mathbf{V}, \mathbf{E})$ if the following holds for $\mathbf{Y}, \mathbf{Z}, \mathbf{W} \subseteq \mathbf{V}$:

1. For $\mathbf{I}_i \in \mathcal{I}$: $P_i(\mathbf{y}|\mathbf{w}, \mathbf{z}) = P_i(\mathbf{y}|\mathbf{w})$ if $\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{W}$ in D_{aug}
2. For $\mathbf{I}_i, \mathbf{I}_j \in \mathcal{I}$: $P_i(\mathbf{y}|\mathbf{w}) = P_j(\mathbf{y}|\mathbf{w})$ if $\mathbf{Y} \perp\!\!\!\perp \mathbf{K}|\mathbf{W}_\mathbf{K}$ in $D_{aug_{\underline{\mathbf{W}_\mathbf{K}}, \overline{\mathbf{R}(\mathbf{W})}}}$

, where $\mathbf{K} := (\mathbf{I_i} \setminus \mathbf{I}_j) \cup (\mathbf{I_j} \setminus \mathbf{I}_i)$, $\mathbf{W}_\mathbf{K} := \mathbf{W} \cap \mathbf{K}, \mathbf{R} := \mathbf{K} \setminus \mathbf{W}_\mathbf{K}$, and $\mathbf{R}(\mathbf{W}) \subseteq \mathbf{R}$ are non-ancestors of $\mathbf{W}$ in D_{aug}.

Assumption A.7 (c-faithfulness)**.** A tuple of distributions $\mathbf{P}$ are said to be *c-faithful* to D_{aug} if the converse of each of the Ψ-Markov conditions holds.

B RELATED WORK

Root Cause Analysis in Microservices. Root Cause Analysis (RCA) is done both online [Wang et al., 2023a] and offline [Deng and Hooi, 2021], often relying on system dependency graphs [Chen et al., 2014]. Previous approaches have used statistical techniques, deep neural networks, and graph representation [Brandón et al., 2020, Capozzoli et al., 2015, Ma et al., 2020]. For instance, [Lin et al., 2018] uses z-scores to compare the distributions of normal operation and anomalous system data. The method finds the root cause by identifying nodes that deviate the most between two distributions, but it imposes normality assumptions on the data and it is sensitive to outliers. Li et al. [2022] also uses similar techniques with a call graph provided by expert knowledge to adjust the scores. Pham et al. [2024] improves this idea by using median and interquartile range instead, but the method does not leverage any causal knowledge. Wang et al. [2023b] used both individual and topological time series data to capture interdependencies between microservices, while Xin et al. [2023] introduced a gradient-based causal structure learning method to generate weighted causal graphs and developed a root cause inference method called CausalRCA. Strobl and Lasko [2023] assume an invertible non-linear SCM with additive noise terms such as non-Gasussian error terms and quantify the root cause contributions using Shapley values based on conditional distributions on the error terms. Strobl [2024] later extends it to settings where counterfactual distributions can be derived. Budhathoki et al. [2022] assume a causal graph with fully specified functional relationships to quantify the contribution of each variable to the target outlier score. Budhathoki et al. [2022] define a root cause to be the variable where its value is being detected to be an outlier among all the values that are jointly observed by other variables. Later, Okati et al. [2024] extends it by providing a more efficient method when the causal graph is known and a heuristic when the causal graph is unknown under the single root cause assumption. Okati et al. [2024] also assumes there is only a single observation available in the anomalous regime. Assaad et al. [2023] uses a fully oriented acyclic summary causal graph with loops learned from time-series data or provided by experts. They impose linearity for each directed edge in the given graph. Recently, Lin et al. [2024] proposed RUN, a method that forecasts time series by constructing a neural network for each system metric and then uses the forecasted data to build a Granger causal graph. During the diagnosis stage, RUN, like other algorithms, applies a weighted personalized PageRank algorithm to traverse the graph and identify the root cause. A closely related work to ours is RCD [Ikram et al., 2022], where Ikram et al. [2022] presented a causal framework that treats failure as an intervention. They developed a hierarchical approach to causal discovery by randomly partitioning the set of observed variables and using a series CI tests in each partition to produce a set of potential root causes. This approach is particularly relevant to our work, as it also employs CI tests to localize and pinpoint the failure's root cause. However, despite the innovative contributions of these recent studies, we argue that a critical aspect has been overlooked: the opportunity to utilize normal operation periods to develop a more efficient and effective RCA method for failure periods.

Causal Discovery with Background Knowledge for RCA. One common approach to learning the causal structure is to incorporate expert knowledge [Chakraborty et al., 2023, Gong et al., 2024, Lin et al., 2024, Xin et al., 2023]. However, it may not always be feasible to obtain expert knowledge. A data-driven approach to causal structure learning then becomes a more viable solution. A key point is that learning causal structures does not require interventional data [Spirtes et al., 2000, Chickering, 2002, Shimizu et al., 2006, Zheng et al., 2018]. We can leverage the vast amounts of observed data generated during the system's normal operation to construct the causal graph, rather than waiting for a failure.

Causal Discovery with Observational Data. A wide class of causal discovery algorithms that learn a CPDAG is mainly score-based or constraint-based Chickering [2002], Spirtes et al. [2000]. For score-based methods, learning a causal structure can be extremely time-consuming [Chickering et al., 2004]. Fortunately, there are recent advances that speed up the processing of constructing an essential graph in the score-based methods [Chickering, 2020, Ramsey et al., 2017, Nazaret and Blei, 2024]. Lam et al. [2022] also provide an efficient algorithm named GRaSP to exploit permutation reasoning to search for a causal graph that is guaranteed to be in the Markov equivalence class of the ground truth. Andrews et al. [2023] extends GRaSP by using Grow-Shrink Tree to make the algorithm more accurate and scalable. With further assumptions, Montagna et al. [2023] also provides a scalable causal discovery algorithm based on score matching to recover a causal graph.

Given that the use of CI tests is a central aspect of our work, we also provide a brief overview of recent advances in causal discovery, particularly those focused on using CI tests. Causal discovery often relies on a series of CI tests to determine relationships between variables. However, this approach can be problematic, as the statistical power of CI tests diminishes with a finite sample size or when the conditioning set is large [Shah and Peters, 2020]. Also, they often involve conditioning on large sets of nodes to identify possible separating sets for each node [Spirtes et al., 2000]. This time-consuming aspect of causal discovery is particularly undesirable in our context, where time is critical following a failure, and the goal is to quickly pinpoint the root cause. A promising direction in addressing this issue has been the exploration of methods to restrict the size of the conditioning set. In the absence of latent confounders, Wienöbst and Liskiewicz [2020] introduced a sound and complete algorithm known as Low-Order Causal Inference (LOCI), which learns a graphical representation based on CI relations of order k or lower. Similarly, Kocaoglu [2023] provided a novel characterization of the graphical representation termed the k-essential graph, along with a sound learning algorithm to construct it. Building on these ideas, Lee et al. [2024] proposed an approach that further restricts the conditioning sets for all CI tests so long these tests include all marginal tests. Our work integrates these recent advancements to develop and utilize a more robust causal graph than the current state-of-the-art in RCA literature.

C THEOREMS AND PROOFS

For the sack clarity, we first provide the Theorem 1 from Shangqi et al. [2023] and Theorem 2 from Tao et al. [2019]. Shangqi et al. [2023] term the IGS problem as the POMS problem and they refer to a DAG as an input graph.

Theorem C.1. *Shangqi et al. [2023] For the POMS problem, let n represent the number of vertices in the input graph D and d denote the maximum vertex out-degree in D. Both of the following statements are true:*

- *There is an algorithm that can find the target in $O(\log_{1+k} n + (d/k)\log_{1+d} n)$ probs.*
- *Any POMS algorithm must perform $\Omega(\log_{1+k} n + (d/k)\log_{1+d} n)$ probs to find the target in the worse case.*

Theorem C.2. *Tao et al. [2019] Let h be the length of the longest path in the DAG the and n be the number of variables. Both of the following statements are true about the IGS problem:*

- `DFS-interleave` *asks at most $\lceil \log_2 h \rceil \cdot (1 + \lfloor \log_2 n \rfloor) + (d-1) \cdot \lceil \log_d h \rceil$ questions.*
- *Any algorithm must ask at least $(d-1) \cdot \lfloor \log_d h \rfloor$ questions in the worst case.*

We provide the pseudocode of `DFS-interleave`, which has been modifed for RCA, in Algorithm 4.

Lemma 4.1. *Given a DAG D, if $(F \perp\!\!\!\perp X)_P$ for some $X \in \mathbf{V}$, then $A \notin Ch_{D_{aug}}(F)$ for all $A \in An_D(X)$, where P is any joint distribution between variables on D_{aug}.*

Proof. For the sake of contradiction, suppose $F \to A$ in D_{aug} for some $A \in An_D(X)$. Since A is an ancestor of X in D, there must be a directed path q from A to X in D. Thus, q must also exist in D_{aug}. Consider the path obtained by concatenating $F \to A$ with q in D_{aug}. This path must be d-connecting in D_{aug}. Thus, it must be that $(F \not\perp\!\!\!\perp X)_{D_{aug}}$. From interventional faithfulness, we have that $(F \not\perp\!\!\!\perp X)_P$, which is a contradiction. □

Lemma 4.2. *Given a DAG D, if $(F \not\perp\!\!\!\perp X)_P$ for some $X \in \mathbf{V}$, then then $Q \notin Ch_{D_{aug}}(F)$ for all $Q \in NAn_D(X)$, where P is any joint distribution between variables on D_{aug}.*

Proof. For the sake of contradiction, suppose $F \to Q$ in D_{aug} for some $Q \in NAn_D(X)$. Since Q is a non-ancestor of X in D, without loss of generality, there are several cases: (i) there exists a directed path q from X to Q in G (ii) there is no path between Q and X in D and (iii) any path p between X and Q must have a collider on p in D.

For case (i), q must also exist and be directed in D. By concatenating the path from X to Q and $F \to Q$, we see the path from F to X is blocked. Thus, we have $(F \perp\!\!\!\perp X)_D$, which implies $(F \perp\!\!\!\perp X)_P$ by Assumption A.6, which is a contradiction.

For case (ii), there is no path between X and Q in D, which implies $(F \perp\!\!\!\perp X)_D$ so that we reach the same contradiction.

For case (iii), every collider on any path p between Q and X must also be in D such that we have $(F \perp\!\!\!\perp X)_D$ by concatenating $F \to Q$ with p, which implies $(F \perp\!\!\!\perp X)_P$ by Assumption A.6, which is a contradiction. □

Lemma 4.3. *Consider a DAG $D = (\mathbf{V}, \mathbf{E})$ with a single sink node and D' be a DAG by reversing every edge direction in $\varepsilon_k()$, let $Q(X)$ be a query to the oracle on whether some $X \in \mathbf{V}$ has a directed path to an unknown target node $R \in \mathbf{V}$.*

$$Q(X) = \textit{yes} \Leftrightarrow (F \not\perp\!\!\!\perp X)_P \tag{1}$$

Therefore, if $Q(X) =$ yes, then $X \in An_{D'}(R)$. If $Q(X) =$ no, then $X \in NA_{D'}(R)$.

Proof. Consider some nodes $X \in \mathbf{V}$, suppose $(F \perp\!\!\!\perp X)_P$, then $X \in NDe_D(R)$ by Lemma 4.1. Note that $NDe_D(R) = NAn_{D'}(R)$ due to $De_D(R) = An_{D'}(R)$ by the given conditions for D and D'. Therefore, $X \in NAn_{D'}(R)$. As $NAn_{D'}(R) \Leftrightarrow Q(X)$= no. We have that $(F \perp\!\!\!\perp X)_P \Rightarrow Q(X) =$ no. Similarly, suppose $(F \not\perp\!\!\!\perp X)_P$, then $X \in De_D(R)$ by Lemma 4.2. As $De_D(R) = An_{D'}(R)$, we have that $(F \not\perp\!\!\!\perp X)_P \Rightarrow X \in An_{D'}(R)$, which is equivalent to $Q(X) =$ yes. □

Theorem 4.4. *Given a DAG D with a single sink node, any algorithm the only uses marginal invariance tests must perform $\Omega(\log_2 n + d \log_{1+d} n)$ many tests to find the single root cause in the worst case, where d is the maximum in-degree of D and n is the number of nodes. There exists an algorithm that finds the root cause with $\mathcal{O}(\log_2 n + d \log_{1+d} n)$ marginal invariance tests.*

Proof. This follows from Lemma 4.3 and Theorem 1 in [Shangqi et al., 2023], which says that any algorithm must ask $\Omega(\log_2 n + d \log_{1+d} n)$ queries to identify the target node selected by an adversary in a DAG D' with a single root node for the problem of IGS, where d is the maximum out-degree in D' and there is an algorithm that can find the target node in $\mathcal{O}(\log_2 n + d \log_{1+d} n)$ number of queries. □

C.1 PROOF OF THEOREM 5.1

We first leverage an existing result from Wienöbst and Liskiewicz [2020] (see Lemma C.3). Then, we will prove Lemma C.4. While Lemma C.3 ensures the correctness of lines 5-6 in Algorithm 1, Lemma C.4 proves the correctness of using possible parent sets to rank the top root causes under a more fine-grained representation of CPDAGs due to the orientations that take place in lines 6-7 in Algorithm 1.

Lemma C.3. *Wienöbst and Liskiewicz [2020] Given a distribution P defined over a DAG D, for any $X, Y \in \mathbf{V}$ and $\mathbf{Z} \in \mathbf{V} \setminus \{X, Y\}, |\mathbf{Z}| \leq k$ for some $k \geq 0$, if $(X \perp\!\!\!\perp Y \,|\mathbf{Z})_P, (X \not\perp\!\!\!\perp W \,|\mathbf{Z})_P$, then no DAG faithful to P contains the edge $W \rightarrow Y$.*

Lemma C.4. *Let M be the graph returned by lines 6-7 in Algorithm 1, F is not adjacent to X in D_{aug} if and only if F is d-separated with X given $PossPa_M(X)$ in D_{aug}.*

Proof. We first prove the if (⇒) direction.

We first give a critical insight. We note that if F-NODE points to any variable that is a collider H on some paths p in D_{aug}, then running marginal tests must have allowed us to orient $F \rightarrow H \leftarrow U$ and $F \rightarrow H \leftarrow Q$ for some variables U, Q on p in the given CPDAG $\mathcal{C}(D)$ due to Lemma C.3. Thus, we call this resulting graph M rather than $\mathcal{C}(D)$. If F is marginally independent with all members in the adjacency set of H, then the result follows.

Suppose there is more than one node being marginally dependent on F. We call this set $\mathbf{Z}$. Then, we know F must have a directed path to all such nodes $Z \in \mathbf{Z}$ in D_{aug} as there is no incoming edges to F and each of these nodes is marginally dependent with F. We will prove the claim that if F is not adjacent to Z in D_{aug}, then F is d-separated with Z given $PossPa_M(Z)$ in D_{aug} for all $Z \in \mathbf{Z}$.

For the sake of contradiction, assume that F is d-connecting with Z given $PossPa_M(Z)$ in D_{aug}. First, we note that $PossPa_M(Z)$ must contain all parents of Z in D_{aug}. Since there exists a directed path from F to Z, we call this path r as shown below:

$$F \rightarrow T \rightarrow \ldots \rightarrow W \rightarrow ... \rightarrow Z. \tag{2}$$

Since $PossPa_M(Z)$ must contain all parents of Z, we consider two cases: (i) conditioning on $PossPa(Z)$ opens an active path from F to Z through the backdoor of some ancestors of $PossPa(Z)$ e.g., W by concatenating a subpath of r as follows :

$$F \rightarrow T \rightarrow \ldots \rightarrow W \leftarrow Q \rightarrow \ldots \rightarrow Z \tag{3}$$

and case (ii): there exists a d-connecting path from F to Z given some variables K as follows

$$F \rightarrow T \rightarrow \ldots \rightarrow W \rightarrow \ldots \rightarrow K \leftarrow Z \tag{4}$$

Case (i) - conditioning on $PossPa(Z)$ opens an active path from F to Z through the backdoor of some ancestors of $PossPa(Z)$ by concatenating with a subpath of r: We will first show a contradiction in this case. Note that we cannot have

$Q \in An_{D_{aug}}(Z)$. To see that, suppose Q and Z is adjacent in $\mathcal{C}(D)$, then Q must be in $PossPa_M(Z)$ as $(F \not\perp\!\!\!\perp Z)_P$ so that Algorithm 1 will not change the orientation of this edge in M. This yields a contradiction as $Q \in PossPa_M(Z)$ would have blocked this path. Suppose they are not adjacent in $\mathcal{C}(D)$, as any directed path from Q to Z would have been blocked by conditioning on $PossPa_M(Z)$, there exists a collider U_1 on a path from Q to Z with a member in $De_{D_{aug}}(U_1)$ must be in $PossPa_M(Z)$ in order for the path in (8) to be a d-connecting path from F to Z as follows.

$$F \to T \to \ldots \to W \leftarrow Q \to \ldots \to U_1 \leftarrow \ldots Z \tag{5}$$

Consider U_1 is a descendant of Z. This means there exists another backdoor path being active by conditioning on $PossPa_M(Z)$ where some members $\mathbf{J} \subseteq PossPa_M(Z)$ are descendants of U_1. We first note that U_1 cannot have a directed path to Z due to acyclicty. That implies $\mathbf{J}$ must be children of Z in D_{aug}. Note that there cannot be any descendant R of U_1 that forms an unshielded collider with any member in $\mathbf{J}$ because it will orient $Z \to C$ for any such child C in $\mathbf{J}$ such that they will not be in $PossPa_M(Z)$. Thus, for any J in $\mathbf{J}$ and some descendants R of U_1, we must have shielded colliders $\langle R, J, Z\rangle$. This implies that U_1 must be adjacent to Z, which implies that U_1 is a child of Z. It also implies that the backdoor path in (10) is active due to conditioning on a child of U_1, which is also in $PossPa_M(Z)$. We call this child H. Consider the variable U_2 that is closest to U_1 on the path in (10) e.g.

$$F \to T \to \ldots \to W \leftarrow Q \to \ldots \to U_2 \to U_1 \leftarrow Z \tag{6}$$

Suppose U_2 is adjacent to Z in D_{aug}, if $U_2 \leftarrow Z$ is in D_{aug}, then we can use the same argument by treating U_2 as U_1 repeatedly until we reach Q to be the closest node such that we reach the conclusion that conditioning on Q would have blocked the backdoor path to reach the same contradiction. If $U_2 \to Z$ is in D_{aug}, then U_2 is in $PossPa_M(Z)$ such that the backdoor path would not be active either by conditioning on $PossPa_M(Z)$. Thus, Z and U_2 cannot be adjacent in D_{aug}. Suppose U_2 is not adjacent to H in D_{aug}, then $U_2 \to U_1 \to H$ must have been oriented in $\mathcal{C}(D)$ by first Meek rule such that $Z \to H$ is also oriented in $\mathcal{C}(D)$ due to acyclicity, which leads to a contradiction as $H \in PossPa_M(Z)$. Thus, U_2 is adjacent to H in D_{aug}. This implies that $U_2 \to H$ is in D_{aug} due to acyclicity. Since Z and U_2 cannot be adjacent in D_{aug}, then $U_2 \to H \leftarrow Z$ must have been oriented in $\mathcal{C}(D)$ such that $U_2 \to H \leftarrow Z$ is also in M. Then, H will not be in $PossPa_M(Z)$, which is a contradiction.

Case (ii): Consider the path in (9). We will use a similar argument we made in case (i). That is, there cannot be any descendant R of W that forms an unshielded collider with any member C in $PossPa_M(Z)$ because it will orient $Z \to C$ for any such child C such that they will not be in $PossPa_M(Z)$. Thus, for the variable J that is the descendant closest to W and is a child of Z must form a shielded collider e.g. $\langle W, J, Z\rangle$. However, since W has a directed path to Z, that implies $W \to Z$ must also be in D_{aug} such that $W \in PossPa_M(Z)$, which leads to a contradiction as the path (9) is no longer d-connecting.

For the only if direction, for the sake of contradiction, assume F and X are adjacent in D_{aug}. Since F and X are d-separated given the possible parents set of X in M, then there is no d-connecting path from F to X given the possible parents set of X, which is a contradiction as F is adjacent to X. □

Theorem 5.1. *Given a CPDAG output by any sound causal discovery algorithms and under causal sufficiency and the extended faithfulness assumption, Algorithm 1 returns the true root cause variables.*

Proof. The proof is based on Lemmas C.3 and C.4. Lemma C.3 proves the correctness of the orientation rules in Algorithm 1 to refine the given CPDAG. Lemma C.4 proves the correctness of lines 10-13 in Algorithm 1 where we use the highest CMI value e.g. $I(F; X|PossPa_M(X))$ to identify the true root causes. Specifically, Assumption A.7 ensures that the non-root cause $\bar{R}$ will have low CMI value e.g. $I(F; \bar{R}|PossPa_M(\bar{R}))$ relative to the true root causes R e.g. $I(F; R|PossPa_M(R)) > I(F; \bar{R}|PossPa_M(\bar{R}))$. □

D DISCUSSION ON INCORPORATING OTHER PARTIAL CAUSAL STRUCTURES

In this section, we will briefly introduce what other partial causal structures can be incorporated into our Algorithm 1. Then, we will discuss the main differences between these structures and a CPDAG. These structure are called k-CPDAG Wienöbst and Liskiewicz [2020] and k-essential graphs Kocaoglu [2023]. These graphical representation are mainly for characterizing a set of causal graphs that share the same d-separation constraints with a bounding conditioning set size up to k e.g. $(X \perp\!\!\!\perp Y|\mathbf{Z})_D, |\mathbf{Z}| \leq k$ for some causal graphs D. For the sake of clarity, we call these *degree-k d-separation*

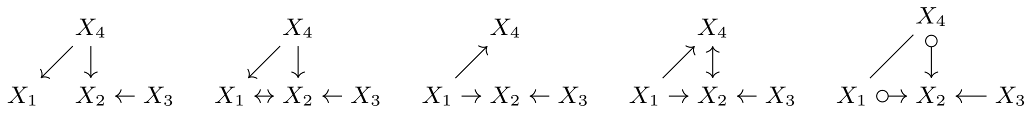

(a) D_1 (b) $\mathcal{C}_k(D_1)$ (c) D_2 (d) $\mathcal{C}_k(D_2)$ (e) ε_k

Figure 5: Kocaoglu [2023] Two k-Markov equivalent DAGs for $k = 0$ with the same k-essential graph. As $\mathcal{C}_k(D_1)$ is Markov equivalent to $\mathcal{C}_k(D_2)$, they have the same k-essential graphs $\varepsilon_k(D_1) = \varepsilon_k(D_2) = \varepsilon_k$, obtained as the edge union of their k-closures. (b): the k-closure graph of D_1, where $k = 0$. (d): the k-closure graph of D_2, where $k = 0$. (e): the k-essential graph of the two k-closures graphs $\mathcal{C}_k(D_1), \mathcal{C}_k(D_2)$.

statements. As k-essential graphs carry strictly more information about the set of causal graphs that entail the set of degree-k d-separation statements than k-CPDAG (see Section D.6 in Kocaoglu [2023]), any result that we extend to k-essential graphs in this section will also be applicable to k-CPDAG. Thus, our discussion will focus on k-essential graphs for the rest of this section. These partial causal structures are developed for practical scenarios where there is only very limited sample, which can quickly compromise the reliability of CI tests due to the lack of statistical power. Kocaoglu [2023] has provided an algorithm known as k-PC for learning k-essential graphs from observed data. By letting $k = |\mathbf{V}| - 2$, a k-essential graph will resemble a CPDAG. Thus, one can view it as a more general form of the CPDAG.

D.1 BACKGROUND ON k-ESSENTIAL GRAPHS

For readability, we provide the definitions and theorems from Kocaoglu [2023] that are necessary to facilitate the discussion. For details, please refer to Kocaoglu [2023]. We begin by introducing several key concepts: k-covered, k-closure graphs, and their relationships with k-essential graphs.

Definition D.1. Kocaoglu [2023] Given a DAG $D = (\mathbf{V}, \mathbf{E})$ and an integer k, a pair of nodes X, Y are said to be k-covered if $\nexists \mathbf{Z} \subset \mathbf{V} : |\mathbf{Z}| \leq k$ and $(X \perp\!\!\!\perp Y \,|\mathbf{Z})_D$.

Definition D.1 says that two variables are k-covered if there is no separating set of size up to k that can d-separate them. The idea of having two nodes being k-covered is to facilitate the construction of an augmented graphical representation of a DAG known as k-closure graphs.

Definition D.2. Kocaoglu [2023] Given a DAG $D = (\mathbf{V}, \mathbf{E})$ and an integer k, the k-closure of D is defined as the graph shown by $\mathcal{C}_k(D)$ that satisfies the following:

1. If: X, Y are k-covered in D
   $i)$ if $X \in An_D(Y)$, then $X \to Y$ in $\mathcal{C}_k(D)$, $ii)$ if $Y \in An_D(X)$, then $X \leftarrow Y$ in $\mathcal{C}_k(D)$,
   $iii)$ else $X \leftrightarrow Y$ in $\mathcal{C}_k(D)$
2. Else: X, Y are non-adjacent in $\mathcal{C}_k(D)$

Kocaoglu [2023] provides a graphical way to help determine whether two k-closure graphs are Markov equivalent. Kocaoglu [2023] also proved that the k-closure graph of a DAG D induces the same set of degree-k d-separation statements as D.

Corollary D.3. *Kocaoglu [2023] Two k-closure graphs K_1, K_2 are Markov equivalent if and only if*

1. *They have the same skeleton and*
2. *They have the same unshielded colliders.*

Lemma D.4. *Kocaoglu [2023] k-closure $\mathcal{C}_k(D)$ of a DAG D entails the same degree-k d-separation statements as the DAG, i.e.,* $(X \perp\!\!\!\perp Y \,|\mathbf{Z})_D \iff (X \perp\!\!\!\perp Y \,|\mathbf{Z})_{\mathcal{C}_k(D)}, \forall \mathbf{Z} \subset \mathbf{V} : |\mathbf{Z}| \leq k$.

Moreover, k-Markov equivalence of two DAGs can be reduced to checking Markov equivalence of their k- closure graphs. Based on this result, Kocaoglu [2023] shows that one can only hope to learn up to the equivalence class of k-closure graphs just by CI tests.

Theorem D.5. *Kocaoglu [2023] Two DAGs D_1, D_2 are k-Markov equivalent if and only if $\mathcal{C}_k(D_1)$ and $\mathcal{C}_k(D_2)$ are Markov equivalent.*

Kocaoglu [2023] uses the following edge union to represent a set of Markov equivalent k-closure graphs.

Definition D.6 (edge unions: —, o—o, $o\!\!\rightarrow$ [Kocaoglu, 2023])**.** The edge union operations of a set of k-closure graphs are defined as: (i) $X — Y := X \to Y \cup X \leftarrow Y$, (ii) $X\ o—o\, Y := X \to Y \cup X \leftarrow Y \cup X \leftrightarrow Y$, (iii) $X\ o\!\!\rightarrow Y := X \to Y \cup X \leftrightarrow Y$. A wildcard mark $*$ denotes it can be a circle, a tail, or an arrowhead mark.

Definition D.7 (k-essential graph)**.** Kocaoglu [2023] For any DAG D, the edge union of all k-closure graphs that are Markov equivalent to $\mathcal{C}_k(D)$ is called the k-essential graph of D, shown by $\varepsilon_k(D)$.

Figure 5 illustrates the difference between DAGs, k-closure graphs, and k-essential graphs.

D.2 THEOREMS AND PROOFS FOR EXTENDING OUR RESULTS TO k-ESSENTIAL GRAPHS

We will first prove a result that is similar to C.3 for k-essential graphs. We will leverage an existing result from Wienöbst and Liskiewicz [2020].

Lemma D.8. *Wienöbst and Liskiewicz [2020] Given a distribution P defined over a DAG D, for any $X, Y \in \mathbf{V}$ and $\mathbf{Z} \in \mathbf{V} \setminus \{X, Y\}, |\mathbf{Z}| \leq k$ for some $k \geq 0$, if $(X \perp\!\!\!\perp Y\,|\mathbf{Z})_P, (X \not\perp\!\!\!\perp W\,|\mathbf{Z})_P, (W \not\perp\!\!\!\perp Y\,|\mathbf{Z})_P$, then no DAG k-faithful to P contains a causal path from W to Y.*

Lemma D.9. *Given a distribution P defined over a DAG D, for any $X, Y \in \mathbf{V}$ and $\mathbf{Z} \in \mathbf{V} \setminus \{X, Y\}, |\mathbf{Z}| \leq k$ for some $k \geq 0$, if $(X \perp\!\!\!\perp Y\,|\mathbf{Z})_P, (X \not\perp\!\!\!\perp W\,|\mathbf{Z})_P$, then the k-essential graph $\varepsilon_k(D)$ of all k-closure graphs that are Markov equivalent to $\mathcal{C}_k(D)$ does not contain $W \to Y$.*

Proof. By design of k-essential graphs $\varepsilon_k(D)$, two nodes X, Y are adjacent in $\varepsilon_k(D)$ only when $(X \not\perp\!\!\!\perp Y|\mathbf{Z})_D$ for all $\mathbf{Z} \subseteq \mathbf{V}, |\mathbf{Z}| \leq k$. It is because by design, there exists at least one k-closure graph that is Markov equivalent to $\mathcal{C}_k(D)$ where X and Y are adjacent, which implies X and Y are k-covered in D. Given that $(X \perp\!\!\!\perp Y\,|\mathbf{Z})_P, (X \not\perp\!\!\!\perp W\,|\mathbf{Z})_P$, we have that $\varepsilon_k(D)$ does not contain $W \to Y$ by Lemma D.8. □

Next, we will prove a result for k-essential graphs that is similar to Lemma C.4 to show the correctness of conditioning on the possible parents in the more fine-grained k-essential graph in Algorithm 2.

Lemma D.10. *Let M be the graph returned by lines 6-9 in Algorithm 2, F is not adjacent to X in D_{aug} if and only if F is d-separated with X given $PossPa_M(X)$ in D_{aug}.*

Proof. We first prove the if ($\Rightarrow$) direction.

We first give a critical insight. We note that if F-NODE points to any variable that is a collider H on some paths p in D_{aug}, then running marginal tests must have allowed us to orient $F {*\!\!\rightarrow} H \leftarrow\!\!* U$ and $F {*\!\!\rightarrow} H \leftarrow\!\!* Q$ for some variables U, Q on p in the given k-essential graphs $\varepsilon_k(D)$ due to Lemma D.9. Thus, we call this resulting graph M rather than $\varepsilon_k(D)$. If F is marginally independent with all members in the adjacency set of H, then the result follows.

Suppose there is more than one node being marginally dependent on F. We call this set $\mathbf{Z}$. Then, we know F must have a directed path to all such nodes $Z \in \mathbf{Z}$ in D_{aug} as there is no incoming edges to F and each of these nodes is marginally dependent with F. We will prove the claim that if F is not adjacent to Z in D_{aug}, then F is d-separated with Z given $PossPa_M(Z)$ in D_{aug} for all $Z \in \mathbf{Z}$.

For the sake of contradiction, assume that F is d-connecting with Z given $PossPa_M(Z)$ in D_{aug}. First, we note that $PossPa_M(Z)$ must contain all parents of Z in D_{aug}. Since there exists a directed path from F to Z, we call this path r as shown below:

$$F \to T \to \ldots \to W \to ... \to Z. \tag{7}$$

Since $PossPa_M(Z)$ must contain all parents of Z, we consider two cases: (i) there exists a backdoor active path from F to Z by concatenating with a subpath of r as follows:

$$F \to T \to \ldots \to W \leftarrow Q \to \ldots \to Z \tag{8}$$

and case (ii): there exists a d-connecting path from F to Z given some variables K as follows

$$F \to T \to \ldots \to W \to \ldots \to K \leftarrow Z \tag{9}$$

Case (i) - there exists a backdoor active path from F to Z by concatenating with a subpath of r: We will first show a contradiction in this case. Note that we cannot have $Q \in An_{D_{aug}}(Z)$. To see that, suppose Q and Z is adjacent in $\varepsilon_k(D)$, then Q must be in $PossPa_M(Z)$ as $(F \not\perp\!\!\!\perp Z)_P$ so that Algorithm 2 will not change the orientation of this edge in M. This yields a contradiction as $Q \in PossPa_M(Z)$ would have blocked this path. Suppose they are not adjacent in $\varepsilon_k(D)$, as any directed path from Q to Z would have been blocked by conditioning on $PossPa_M(Z)$, there exists a collider U_1 on a path from Q to Z with a member in $De_{D_{aug}}(U_1)$ must be in $PossPa_M(Z)$ in order for the path in (8) to be a d-connecting path from F to Z as follows.

$$F \to T \to \ldots \to W \leftarrow Q \to \ldots \to U_1 \leftarrow \ldots Z \tag{10}$$

Consider U_1 is a descendant of Z. This means there exists another backdoor path being active by conditioning on $PossPa_M(Z)$ where some members $\mathbf{J} \subseteq PossPa_M(Z)$ are descendants of U_1. We first note that U_1 cannot have a directed path to Z in D_{aug} due to acyclicty. That implies $\mathbf{J}$ must be children of Z in D_{aug}. Note that there cannot be any descendant R of U_1 that forms an unshielded collider with any member in $\mathbf{J}$ because it will orient $Z \to C$ for any such child C in $\mathbf{J}$ such that they will not be in $PossPa_M(Z)$. Thus, for any J in $\mathbf{J}$ and some descendants R of U_1, we must have shielded colliders $\langle R, J, Z\rangle$. This implies that U_1 must be adjacent to Z, which implies that $U_1 \leftarrow\!\!* Z$ is in M.

It also implies that the backdoor path in (10) is active due to conditioning on a variable that is adjacent to U_1 and it is in $PossPa_M(Z)$. Let us call this variable H. Consider the variable U_2 that is closest to U_1 on the path in (10) in D_{aug} e.g.

$$F \to T \to \ldots \to W \leftarrow Q \to \ldots \to U_2 \to U_1 \leftarrow Z \tag{11}$$

Suppose U_2 is adjacent to Z in M, if $U_2 \leftarrow\!\!* Z$ is in M, then we can use the same argument by treating U_2 as U_1 repeatedly until we reach Q to be the closest node such that we reach the conclusion that conditioning on Q would have blocked the backdoor path to reach the same contradiction. If $U_2 *\!\!\to Z$ or $U_2 o\!\!-\!\!o Z$ is in M, then U_2 is in $PossPa_M(Z)$ such that the backdoor path would not be active either by conditioning on $PossPa_M(Z)$ in D_{aug}. Thus, Z and U_2 cannot be adjacent in M. Suppose U_2 is not adjacent to H in D_{aug}, if U_2 and H are not k-covered, then $U_2 *\!\!\to U_1 *\!\!\to H$ must have been oriented in $\varepsilon_k(D)$ by first Meek rule in k-PC algorithm Kocaoglu [2023] such that $Z *\!\!\to H$ is also oriented in $\varepsilon_k(D)$ due to acyclicity, which leads to a contradiction as $H \in PossPa_M(Z)$. Thus, U_2 is adjacent to H in M. This trivially also holds if U_2 and H are k-covered. This implies that $U_2 \to H$ is in D_{aug}. Since Z and U_2 cannot be adjacent in M, then $U_2 *\!\!\to H \leftarrow\!\!* Z$ must have been oriented in $\varepsilon_k(D)$ such that $U_2 *\!\!\to H \leftarrow\!\!* Z$ is also in M. Then, H will not be in $PossPa_M(Z)$, which is a contradiction.

Case (ii): Consider the path in (9). We will use a similar argument we made in case (i). That is, there cannot be any descendant R of W that forms an unshielded collider with any member C in $Ch_M(Z)$ because it will orient $Z *\!\!\to C$ for any such child C such that they will not be in $PossPa_M(Z)$. Thus, for the variable J that is the descendant closest to W and is a child of Z must form a shielded collider e.g. $\langle W, J, Z\rangle$ in M. However, since W has a directed path to Z, that implies $W *\!\!\to Z$ must also be in M such that $W \in PossPa_M(Z)$, which leads to a contradiction as the path (9) is no longer d-connecting.

For the only if direction, for the sake of contradiction, assume F and X are adjacent in D_{aug}. Since F and X are d-separated given the possible parents set of X in M, then there is no d-connecting path from F to X given the possible parents set of X, which is a contradiction as F is adjacent to X. □

Theorem D.11. *Given the k-essential graph of the true DAG D and under causal sufficiency and the extended faithfulness assumption, Algorithm 2 returns the true root cause variables.*

Proof. The proof is based on Lemmas D.9 and D.10. Lemma D.9 proves the correctness of the orientation rules in Algorithm 2 to refine the given k-essential graph. Lemma D.10 proves the correctness of lines 16-19 in Algorithm 2 where we use the highest CMI value e.g. $I(F; X|PossPa_M(X))$ to identify the true root causes. Specifically, assumption A.7 ensures us that the non-root cause $\bar{R}$ will have low CMI value e.g. $I(F; \bar{R}|PossPa_M(\bar{R}))$ relative to the true root causes R e.g. $I(F; R|PossPa_M(R)) > I(F; \bar{R}|PossPa_M(\bar{R}))$. □

E ALGORITHMS

Algorithm 2 RCA with Causal Graphs (Extended RCG)

```
input  Observational data $\mathcal{D}$, interventional data $\mathcal{D}^\star$, a $k$-essential graph $\varepsilon_k(D) = (\mathbf{V}, \mathbf{E})$, Max. no of root causes $l$,
output  Top-$l$ root causes
 1: Concatenate $\mathcal{D}$ and $\mathcal{D}^\star$ with a binary indicator variable $F$.
 2: for $X \in \mathbf{V}$ do
 3:     $A_X \leftarrow I(F;X)$
 4: $A \leftarrow$Sort $X \in \mathbf{V}$ by $A_X$ in ascending order
 5: Create an empty list $\mathbf{V}_s^\star$
 6: for $\alpha \in A$ do
 7:     $G \leftarrow \varepsilon_k(D)$
 8:     for $X, Y \in \mathbf{V}$ do
 9:         if $I(F;X) < \alpha$ and $I(F;Y) \geq \alpha$ then
10:             If $X \leftarrow Y$ is in $G$, remove $X \leftarrow Y$
11:             If $X - Y$ is in $G$, orient $X \rightarrow Y$
12:             If $Xo—oY$ is in $G$, orient $Xo\!\rightarrow Y$
13:             If $X \leftarrow\!oY$ is in $G$, orient $X \leftrightarrow Y$
14:     for $X \in \mathbf{V}$ do
15:         $CMI_X \leftarrow I(F;X|PossPa_G(X))$
16:     $\mathbf{V}_s \leftarrow$Sort $X \in \mathbf{V}$ by $CMI_X$ in descending order
17:     if $\exists X$ that has $I(F;X) < \alpha$ and $CMI_X$ is ranked on top-$l$ in $\mathbf{V}_s$ then
18:         Return the first $l$ root causes from $\mathbf{V}_s^\star$.
19:     $\mathbf{V}_s^\star \leftarrow \mathbf{V}_s$
20: Return the first $l$ root causes from $\mathbf{V}_s^\star$.
```

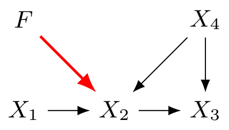
Algorithm 3 CONSTRUCT-HEAVY-PATH-DFS-TREE Tao et al. [2019]

```
input  DAG $D = (\mathbf{V}, \mathbf{E})$
output  A heavy-path-DFS-tree $T$
 1: Create a stack $\mathcal{S}$ with the root node $R$ in $D$ and mark $R$ visited.
 2: repeat
 3:     $J \leftarrow$ get the top member in the stack.
 4:     if $J$ has any child $A$ that has not been visited previously then
 5:         $A' \leftarrow$ Find the child that can reach the highest number of nodes that have not been visited via a directed path.
 6:         Push $A'$ into the stack $\mathcal{S}$ and mark it visited.
 7:     else
 8:         Pop $J$ out of the stack $\mathcal{S}$.
 9: until $\mathcal{S}$ is empty
```

Algorithm 4 Modified IGS (DFS-Interleave Tao et al. [2019]) for RCA

```
input DAG $D = (\mathbf{V}, \mathbf{E})$, interventional data $\mathcal{D}$, CI tester,
output A root cause $R$
 1: if $D$ has more than one sink node then
 2:     $D \leftarrow$ Add a dummy vertex $S$ to $D$ where all the sink nodes in $D$ point to $S$.
 3: $D \leftarrow$ Reverse all the edges in $D$
 4: $T \leftarrow$ CONSTRUCT-HEAVY-PATH-DFS-TREE($D$) {See Algorithm 3}
 5: $\hat{R} \leftarrow$ Select the root of $T$
 6: repeat
 7:     $\pi \leftarrow$ Select the leftmost $\hat{R}$-to-leaf path of $T$
 8:     $U \leftarrow$ Perform binary search on $\pi$ to find the last node $U$ that gives $(F \not\perp\!\!\!\perp U)_P$.
 9:     $W \leftarrow$ Find the leftmost child of $U$ in $T$ where $(F \not\perp\!\!\!\perp W)_P$.
10:     if $W$ does not exists then
11:         return $U$
12:     else
13:         update $\hat{R} \leftarrow W$
14: until $\hat{R}$ has not been updated.
```

F SAMPLE RUN OF RCD Ikram et al. [2022]



$$\begin{tikzcd}
F \arrow[dr] & & X_4 \arrow[dl] \arrow[d] \\
X_1 \arrow[r] & X_2 \arrow[r] & X_3
\end{tikzcd}$$

Figure 6: An example to show how RCD works. RCD would need increase the size of the separating set to 2 to find the root cause (X_2). However, we can leverage the causal graph to know precisely the separating set for every node.

RCD is based on the observation that a failure in a microservice can be treated as an intervention in the underlying causal graph. By treating the root cause as the interventional target, RCD leverages recent advances in causal discovery to identify the root cause. Consistent with the broader causal discovery literature, RCD determines the interventional target (the root cause) through a series of CI tests. RCD operates by introducing a special node, referred to as F, into the dataset and connecting it to every other node in a complete undirected graph. The algorithm's primary goal is to trim down the children of F, as the true root cause will ultimately be the sole remaining child. However, due to the lack of information about the underlying graphical structure, RCD must condition on every possible set of variables until it identifies a separating set that can exclude a potential node as the root cause.

For instance, consider the ground truth causal graph shown in Figure 6, where the root cause is X_2. Initially, RCD constructs an undirected graph with F having outgoing edges to every node. It begins with a separating set of size 0 and executes all possible CI tests. After conducting the tests $(F \perp\!\!\!\perp X_1)_P$ and $(F \perp\!\!\!\perp X4)_P$, RCD removes the edges between F and both X_1 and X_4. At this point, only two candidates for the root cause remain: X_2 and X_3. To narrow it down to the true root cause, RCD increases the size of the separating set. If it tests X_2, it runs $(F \not\perp\!\!\!\perp X2|X_3)_P$. Since X_2 is the root cause, it cannot be independent of F. When testing X_3 by running $(F \not\perp\!\!\!\perp X3|X_2)_P$, conditioning on X_2 opens a backdoor path from F to X_3, preventing its elimination. RCD then increases the size of the separating set once more and runs $(F \perp\!\!\!\perp X3|X_2, X_4)_P$, which removes the edge between F and X_3. Finally, RCD stops, identifying X_2 as the root cause.

Since RCD lacks access to the causal graph, it must perform CI tests on all possible conditioning sets (up to size 2) to identify the root cause, resulting in an exponential growth in tests and higher computational costs. To address this, RCD limits the conditioning set size using a hyperparameter, though this can lead to incomplete results. We propose that knowing the causal graph can significantly reduce the number of required CI tests. A causal graph provides precise separating sets, allowing the root cause to be identified with at most n CI tests, where n corresponds to the number needed for validation of the structure.

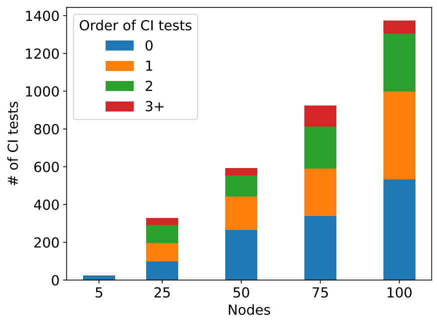


Figure 7: The number of CI tests executed by RCD and the size of the separating set used in those tests. As the number of nodes increases, RCD relies on higher-order CI tests to identify the root cause. However, these higher-order tests are less reliable with limited samples, which diminishes RCD's effectiveness.

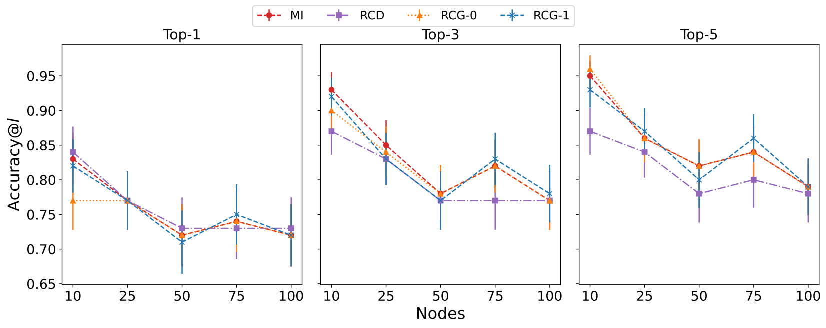


Figure 8: Top-l accuracy and the runtime of extended version of RCG (see Algorithm 2) compared to the baselines. The input graph in this experiment was learned from the data using k-PC.

G ADDITIONAL EXPERIMENTS

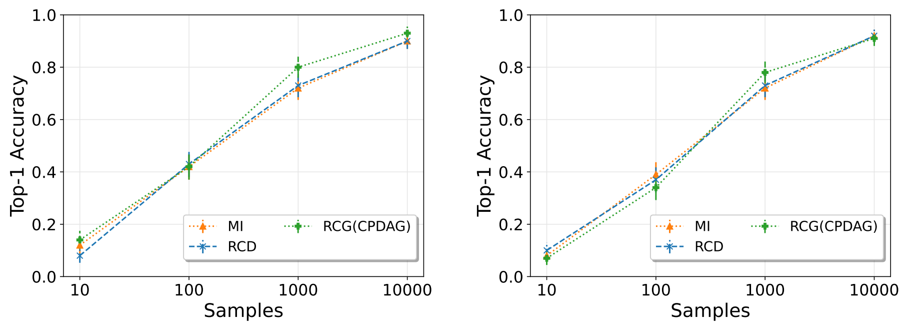
G.1 RCD WITH HIGHER-ORDER CI TESTS

Figure 7 illustrates the number of CI tests executed by RCD alongside the size of the separating sets used. RCD identifies the root cause by gradually increasing the size of these sets. However, the statistical power of CI tests diminishes with larger separating sets, particularly when sample sizes are limited, as is often the case in RCA, where quick failure resolution is crucial [Shah and Peters, 2020, Kocaoglu, 2023]. This reliance on higher-order CI tests leads to poorer performance with an increasing number of nodes, as discussed in Section 6 of the main paper. In contrast, we demonstrate the utility of the extended version of RCG (see Algorithm 2), which mitigates this issue by using k-PC, which is more effective than full graph learning. It relies solely on n marginal invariance tests after failure.

G.2 EXPERIMENTS WITH SAMPLED VERSION

Figure 8 illustrates the performance of RCG in comparison to MI and RCD. Similar to the experiment using the ground truth causal graph, we utilized 10,000 samples for the observational dataset and only 1000 samples for the interventional dataset. Additionally, we included RCG-0 and RCG-1 based on Algorithm 2 to demonstrate the performance across different values of k for k-PC, where uses all the separating set of size up to k. We did not include RUN in this experiment, as it requires continuous data, while our dataset in this experiment is discrete. Furthermore, RCG(IGS) and RCG(CPDAG) were omitted since we cannot derive a complete DAG from the samples, and learning the full CPDAG from the samples is exceedingly time-consuming Ikram et al. [2022].

The results align with our earlier findings presented in the main paper. RCD exhibits poor performance because it lacks access to causal relationships, leading it to condition on all nodes until a separator is found. This results in lower accuracy for RCD. In contrast, RCG yields better results as the value of k increases. Notably, RCG-1 consistently outperforms RCD,



(a) 50-node graphs (b) 100-node graphs

Figure 9: Top-1 accuracy with 50- and 100-nodes graphs with varying number of interventional samples. RCG (CPDAG) (Algorithm 1) uses the ground truth CPDAG.

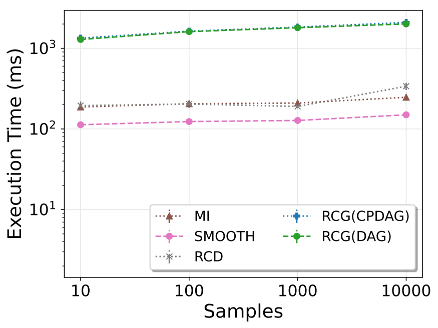


Figure 10: The execution time of ranking top-1 root cause with three competing approaches with varying number of interventional samples.

while RCG-0 occasionally produces results similar to RCD, but sometimes fails to identify the root cause. This inconsistency arises because RCG-0 struggles to learn a sufficiently sparse graph, resulting in conditioning on a larger set of nodes, which diminishes the reliability of the conditional independence test.

G.3 ACCURACY WITH LARGER GRAPHS

Figure 9 presents the top-1 accuracy of the three approaches across 50- and 100-node graphs with varying sample sizes. As we can see the performance of RCG, when graph complexity increases with more nodes, a larger number of samples is required to accurately estimate relationships and identify the root cause.

G.4 EXECUTION TIME WITH VARYING NUMBER OF SAMPLES

Figure 10 presents the execution time of RCG alongside MI and RCD given ground truth CPDAG. MI maintains a consistent runtime as it computes a fixed number of mutual information scores. In contrast, RCD's runtime grows sharply beyond 1000 samples due to an increasing number of dependencies detected by CI tests, leading to a larger number of subsets for analysis. RCG shows the highest but stable runtime. This is because, as RCG has access to more samples, the CMI scores given a separating set become more reliable, enabling RCG to explore different values of α and achieve a more consistent ranking.

H MEMORY LEAK FAILURE IN SOCK-SHOP

		MI	cRCA	RUN	BARO	RCD	RCG-0	RCG-1	RCG-Expert
top-1	Carts	0.87	0.20	0.02	1.00	0.58	1.00	0.87	0.36
	Catalogue	0.10	0.20	0.00	1.00	0.20	0.97	0.12	0.49
	Orders	1.00	0.00	0.00	1.00	1.00	0.95	0.98	0.95
	Payment	0.99	0.40	0.00	1.00	0.93	0.88	0.99	1.00
	User	0.98	0.40	0.00	1.00	1.00	0.44	0.98	0.97
	Avg.	0.79	0.24	0.00	1.00	0.74	0.85	0.79	0.75
top-3	Carts	1.00	0.60	0.40	1.00	0.76	1.00	1.00	1.00
	Catalogue	0.98	0.25	0.30	1.00	0.46	0.99	0.58	1.00
	Orders	1.00	0.00	0.09	1.00	0.96	0.97	0.99	1.00
	Payment	1.00	0.40	0.10	1.00	0.98	0.89	1.00	1.00
	User	1.00	0.62	0.11	1.00	1.00	0.51	1.00	1.00
	Avg.	1.00	0.37	0.20	1.00	0.83	0.87	0.93	1.00
top-5	Carts	1.00	0.80	0.66	1.00	0.77	1.00	1.00	1.00
	Catalogue	0.99	0.52	0.60	1.00	0.49	1.00	0.69	1.00
	Orders	1.00	0.00	0.16	1.00	1.00	0.97	1.00	1.00
	Payment	1.00	0.40	0.19	1.00	0.96	0.89	1.00	1.00
	User	1.00	0.67	0.26	1.00	1.00	0.68	1.00	1.00
	Avg.	1.00	0.48	0.37	1.00	0.84	0.91	0.94	1.00

Table 4: The table presents the top-l accuracy of various baselines on data collected from the Sock-shop application after injecting a memory leak failure into a specific microservice.