EVOCAUSE: LLM-GUIDED EVOLUTION OF CAUSAL GRAPHS FOR ROOT CAUSE ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern telecommunication, cloud, and microservice systems emit correlated alarm cascades when components fail. Root cause analysis (RCA) aims to identify the small set of alarms that initiate each cascade. A common approach learns a causal graph from observational logs and predicts all zero-in-degree alarms in each incident-induced subgraph. However, the learned graph remains fixed and cannot benefit from expert diagnoses of historical incidents. We close this loop with **EvoCause**. Expert labels constrain which alarms should be source nodes but do not specify the edge edits needed to satisfy those constraints. EvoCause uses a large language model (LLM) to propose semantically plausible graph edits, while deterministic code validates node identities and acyclicity and retains the best graph on a labeled alignment set. At test time, the refined graph alone produces transparent predictions without an LLM call. We also release TeleRCA, an expert-annotated benchmark from a production telecommunication network containing $485{,}681$ alarm events spanning 194 alarm types over $5{,}621$ resources. On synthetic data, EvoCause initialized with the PC causal discovery algorithm outperforms the unrefined PC baseline, raising Node F1, Case EM, and Graph F1 by 11.59, 9.40, and 4.59 percentage points, respectively, while reducing nSHD by 0.2379. On TeleRCA, replacing human-readable alarm titles with anonymous identifiers lowers Node F1 and Case EM by 6.12 and 8.04 percentage points, respectively, indicating that alarm-name information contributes to graph refinement.

1 INTRODUCTION

Modern telecommunication, cloud, and microservice systems comprise many tightly coupled components, so a single fault can trigger a cascade of correlated alarms, known as an "alert storm" (Zhao et al., 2020). In this context, root cause analysis (RCA) is the task of identifying, among these alarms, the smallest set of actionable events whose remediation actually resolves the incident (Assaad et al., 2023). Rapid diagnosis is critical because major outages can disrupt many dependent services (O'Brien & Bajak, 2021).

A common causal approach represents alarm types and their triggering relations as a directed acyclic graph (DAG) and predicts all zero-in-degree alarms in the subgraph induced by each incident. This framing provides traceable predictions and generalizes across incidents by encoding triggering relations rather than memorized cases.

Constructing such a graph manually is impractical at scale, so causal discovery algorithms learn it from observational data (Assaad et al., 2022; Spirtes et al., 2000). However, their assumptions and dependence on clean data can produce spurious or missing edges in production alarm streams (Aït-Bachir et al., 2023).

Crucially, existing methods ignore a complementary source of information that could correct these structural errors, namely the historical incident logs curated by Site Reliability Engineering teams, which pair observed alarm sequences with expert-labeled root causes. These diagnostic labels are sparse and expensive to obtain. More importantly, they provide task-level constraints rather than direct edge supervision: they can rule out graphs whose incident-induced source nodes disagree with expert diagnoses, but they generally do not identify a unique ground-truth DAG.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

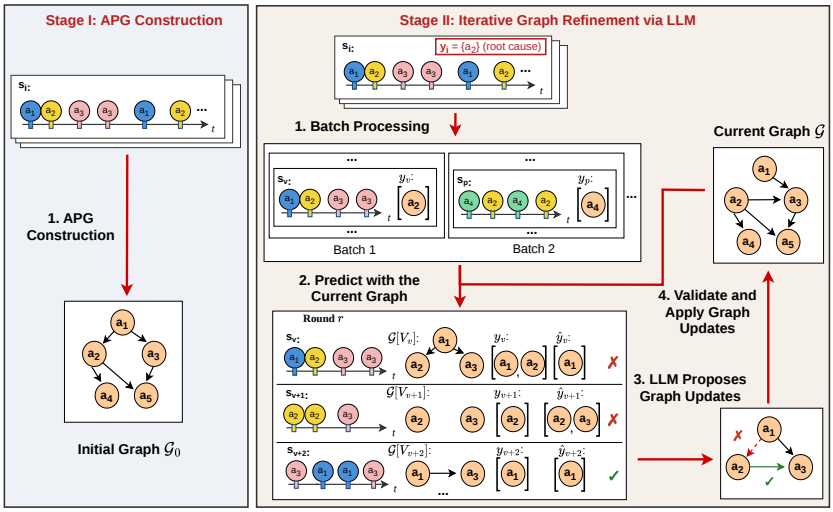


Figure 1: Overview of EvoCause. Stage I learns an initial alarm propagation graph $\mathcal{G}_0$ from historical alarm sequences. Stage II predicts all source nodes in each incident subgraph and uses mismatches with expert labels across a batch to guide LLM proposals for edge additions, removals, or reversals. Deterministic code validates node identities and acyclicity, applies valid edits, and retains the best graph on the alignment set. RCA at test time uses only the selected graph.

This feedback is indirect and underdetermined because expert labels specify which alarms should be sources but not which edge additions, removals, or reversals should enforce them without degrading predictions for overlapping incidents. Language models are useful because they can jointly interpret constraints from multiple incidents, the current graph, optimization history, and, when available, information from alarm titles. We therefore propose **EvoCause** (Evolve Causal Graph), which uses an LLM to rank a small set of semantically plausible edits from this combinatorial space rather than to predict the root cause directly. Deterministic procedures validate every edit, reject cyclic candidates, and retain the graph with the best alignment-set performance. The graph is the only optimized state, whereas its derived topological order is used only as prompt context rather than as a prediction rule. We evaluate the contribution of alarm-title information on TeleRCA by replacing only the titles supplied to the LLM with fixed anonymous identifiers while preserving the graph structure and incident data. Our contributions are summarized as follows:

- On the methodological side, we formulate expert incident labels as *source constraints* on incident subgraphs and characterize the resulting set of root cause consistent graphs. We introduce **EvoCause**, a framework that combines LLM proposals for graph edits, deterministic structural validation, and graph selection using expert feedback. EvoCause supports different causal discovery algorithms without changing their learning procedures and preserves graph validity while reducing disagreement with expert annotations.
- On the data side, we release **TeleRCA**, an RCA benchmark from a production telecommunication network containing 10,922 incidents, 485,681 alarm events, and 194 alarm types across thousands of network resources. TeleRCA combines alarm sequences with expert root cause annotations to support the development and evaluation of RCA methods.
- On the experimental side, experiments on synthetic data and TeleRCA show that EvoCause improves root cause analysis across discovery initializations and graph reconstruction on synthetic data. Further analyses assess the contribution of alarm semantics and sensitivity to the choice of LLM.

2 Related Work

Causal graphs and expert feedback. Observational data generally identify a causal DAG only up to a Markov equivalence class under some standard assumptions. A completed partially di-

rected acyclic graph (CPDAG) represents this class. Directed edges are shared by all compatible DAGs, while undirected edges remain ambiguous (Chickering, 2002). Consistent edge-orientation background knowledge can orient some ambiguous edges and narrow the class (Fang et al., 2025). Propagating all implied orientations yields a maximally oriented partially directed acyclic graph (MPDAG), which represents the DAGs consistent with both the data and the background knowledge (Perković, 2020; Guo & Perković, 2022). Expert root-cause labels instead constrain source sets in incident-induced subgraphs and may require skeleton changes, so EvoCause searches a root-cause-consistent graph set that can span multiple equivalence classes. Interventional equivalence would apply only if incidents were known interventions (Hauser & Bühlmann, 2012). For telecommunication RCA, Zhang et al. (2021) learn a causal graph using Hawkes processes and conditional independence tests, estimate edge weights through causal propagation embeddings, and identify root cause alarms in data streams using influence maximization. THPs (Cai et al., 2024) incorporates network topology into Hawkes processes to learn causal relations among event types. CCCM (Zhang et al., 2024) introduces cluster-aware discovery for base-station alarms, while TTCTH (Li et al., 2025) combines topological–temporal representation learning with a Hawkes-process objective. For microservices, CIRCA (Ikram et al., 2022) treats failures as interventions, RUN (Lin et al., 2024) combines neural Granger causality with contrastive learning, and CausIL (Chakraborty et al., 2023) incorporates domain knowledge into instance-level causal graphs. For threshold-based IT systems, Zan et al. (2024) combine offline causal discovery with online subgraph traversal. APGNN (Jiang & Bai, 2023), Chain-of-Event (Yao et al., 2024), and Groot (Wang et al., 2021) construct event or alarm graphs for diagnosis using learned associations, interpretable edge parameters, or operator rules. These methods provide graph-learning and localization mechanisms but generally do not repeatedly revise an alarm-type DAG using accumulated expert root sets. HRLHF (Wang et al., 2023) does query engineers during dependency-graph discovery, whereas EvoCause uses previously labeled incidents as an offline task signal for refining a graph produced by any discovery backbone.

LLM-based RCA and graph refinement. LLM-based RCA systems have been studied for recommending root causes and mitigations from cloud-incident text (Ahmed et al., 2023). RCACopilot (Chen et al., 2024) combines incident-specific diagnostic collection with classification and explanation. RCAgent (Wang et al., 2024) and ReAct-based systems (Roy et al., 2024) use tools and retrieval, while OpenRCA (Xu et al., 2025) and Flow-of-Action (Pei et al., 2025) provide agent-oriented evaluation or procedure-constrained diagnosis. These approaches mainly predict, explain, or plan actions for an individual incident, so their outputs need not update a shared causal model. Cloud Atlas (Xie et al., 2024) is closer to reusable graph construction because it synthesizes causal graphs from documentation, telemetry, and deployment feedback and then validates them with data. Outside RCA, CAMA refines a mathematical causal graph using question-answer feedback (Zan et al., 2026). EvoCause instead refines an externally discovered alarm graph using expert root-cause labels and performs test-time inference with the refined graph alone. The LLM proposes semantic candidates, while deterministic procedures validate node identities and acyclicity, select the best graph, and perform all test-time inference.

Datasets. The telecom data used by Zhang et al. (2021) provide root cause labels for 6,000 sampled alarm transactions, with annotated causal edges involving 15 of the 78 alarm types. OpenRCA contains 335 software failures and more than 68 GB of logs, metrics, and traces (Xu et al., 2025). In comparison, TeleRCA contains 10,922 incidents with expert root cause annotations, comprising 485,681 alarm events across 119,039 timestamps, 5,621 network resources, and 194 alarm types. It supports RCA evaluation but lacks a ground truth propagation graph for assessing graph reconstruction.

3 EvoCause

This section formulates RCA on alarm cascades and presents EvoCause, which has learning and inference stages. Section 3.1 defines the problem and assumptions, Section 3.2 analyzes identifiability under root-cause feedback, Section 3.3 summarizes the framework, Section 3.4 describes graph construction and label-feedback alignment, and Section 3.5 explains root cause inference.

3.1 PROBLEM SETUP

Dataset Description. We consider a telecommunication network that generates alarms from a finite set of alarm types, denoted by $\mathbf{A}$. The dataset contains m incident sequences, $\mathcal{S} = \{\mathbf{s}_i\}_{i=1}^{m}$, collected over a global observation window $\mathbf{T}$. Each incident is represented by an event sequence $\mathbf{s}_i = \{(a_j, t_j, l_j)\}_{j=1}^{n_i}$, where n_i is the number of alarm events. Each triplet contains an alarm type $a_j \in \mathbf{A}$, a timestamp $t_j \in \mathbf{T}_i$, and an expert label $l_j \in \{0, 1\}$ indicating whether the event is a root cause of the incident. Specifically, $l_j = 1$ indicates a root-cause event, whereas $l_j = 0$ indicates a non-root-cause event. The corresponding incident-level expert root set is $y_i = \{a_j : (a_j, t_j, l_j) \in \mathbf{s}_i,\ l_j = 1\}$, where repeated occurrences of the same alarm type are included only once. We further assume that the triggering relations among alarm types in $\mathbf{A}$ remain invariant throughout the observation window.

Alarm propagation graph. We represent the triggering relations among the alarm types $\mathbf{A}$ with a directed acyclic graph $\mathcal{G} = (\mathbf{A}, \mathbf{E})$, which we call an *alarm propagation graph* (APG). Its nodes are alarm types, and an edge $(a_i, a_j) \in \mathbf{E}$ means that an occurrence of a_i may trigger a_j. For an incident, RCA restricts $\mathcal{G}$ to the observed alarm types and predicts every zero-in-degree node in the induced subgraph as a root cause. The graph is the only persistent structural state. When a linear representation is useful for prompting, we deterministically derive a topological order $O(\mathcal{G}) = \textsc{TopologicalOrder}(\mathcal{G})$, in which a_i precedes a_j for every edge $(a_i, a_j) \in \mathbf{E}$. This order provides a compact linear view of the graph for the LLM. It is derived from $\mathcal{G}$ and is neither optimized nor used for root-cause prediction.

Incident-level source operator. Let $V_i \subseteq \mathbf{A}$ denote the distinct alarm types observed in incident $\mathbf{s}_i$. For a candidate APG $\mathcal{G}$, define $\mathrm{Src}_{\mathcal{G}}(V_i) = \{v \in V_i : \mathrm{Pa}_{\mathcal{G}}(v) \cap V_i = \varnothing\}$. EvoCause predicts the complete set $\mathrm{Src}_{\mathcal{G}}(V_i)$ and neither ranks these nodes nor uses the ground-truth number of roots.

Learning Objective. A conventional causal approach obtains such a graph from the statistical regularities of the observed alarm events alone, without using the expert labels. A causal discovery algorithm $\textsc{CausalDiscovery}(\cdot)$ maps the alarm types and their timestamps to a DAG,

$$\hat{\mathcal{G}} = \textsc{CausalDiscovery}\big(\{(a_j, t_j)\}\big), \tag{1}$$

and root causes are then localized by a procedure $h(\cdot)$ that assigns a binary label to every event,

$$\big[\hat{l}_1, \ldots, \hat{l}_{n_i}\big] \;=\; h\Big(\hat{\mathcal{G}},\, \{(a_j, t_j)\}_{j=1}^{n_i}\Big)\,. \tag{2}$$

In this pipeline, the expert labels l_j never enter the construction of $\hat{\mathcal{G}}$, because the graph is recovered from event statistics alone.

The core objective is to use expert-annotated incidents to refine the graph so that it recovers complete root sets more accurately. We split the incidents into an offline training set $\mathcal{S}_{\text{off}}$ and a held-out test set $\mathcal{S}_{\text{on}}$. For causal discovery, let $\pi_{\text{obs}}(\mathbf{s}_i) = \{(a_j, t_j)\}_{j=1}^{n_i}$ denote the label-free projection of incident $\mathbf{s}_i$, and define $\mathcal{D}_{\text{hist}} = \{\pi_{\text{obs}}(\mathbf{s}_i) : \mathbf{s}_i \in \mathcal{S}_{\text{off}}\}$. The initial graph is learned from $\mathcal{D}_{\text{hist}}$, while the labeled alignment set $\mathcal{D}_{\text{align}}$ is used for graph refinement and model selection. All candidate-graph decisions use $\mathcal{D}_{\text{align}}$, and $\mathcal{S}_{\text{on}}$ is evaluated only once after selection. We select

$$\mathcal{G}^* = \arg\max_{\mathcal{G}} \mathbb{E}_{(\mathbf{s}_i, y_i) \sim \mathcal{D}_{\text{align}}} \left[\mathbf{1}\{\textsc{RCAEngine}(\mathbf{s}_i, \mathcal{G}) = y_i\}\right], \tag{3}$$

where y_i is the complete expert root set and $\mathbf{1}\{\cdot\}$ is the indicator function.

Root-cause-consistent graph set. Let $\mathcal{D}_L = \{(V_i, y_i)\}_{i=1}^{m}$ denote a set of labeled incidents, where $y_i \subseteq V_i$ is the expert-provided root set for incident i. We define the *root-cause-consistent* (RCC) graph set as

$$\mathcal{C}_{\text{RCC}}(\mathcal{D}_L) = \{\mathcal{G} \in \mathrm{DAG}(\mathbf{A}) : \mathrm{Src}_{\mathcal{G}}(V_i) = y_i,\ \forall i\}\,. \tag{4}$$

More generally, two graphs are RCA-equivalent on $\mathcal{D}_L$ if they produce the same source set for every labeled incident. Unlike a Markov equivalence class, the RCC set may contain graphs with different skeletons and conditional-independence structures. When exact consistency cannot be

achieved because of label noise, a restricted search space, or optimization limitations, we quantify inconsistency using the empirical disagreement

$$\mathcal{L}_{\mathrm{RCA}}(\mathcal{G}, \mathcal{D}_L) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}\{\mathrm{Src}_{\mathcal{G}}(V_i) \neq y_i\}\,. \tag{5}$$

The objective in equation 3 then selects the visited graph with the smallest empirical disagreement, using Node F1 to break ties.

Assumptions. Let $\mathcal{K}$ denote the resource–time contexts and let $\mathcal{G}^{(c)}$ be the alarm-type causal graph in context c.

Assumption 1 (Causal consistency) *A shared DAG $\mathcal{G}^\dagger$ satisfies $\mathcal{G}^{(c)} = \mathcal{G}^\dagger$ for every $c \in \mathcal{K}$, corresponding to causal stationarity across contexts (Assaad et al., 2022).*

Assumption 2 (Causal sufficiency) *No latent alarm event is a common cause of two observed alarm types (Spirtes et al., 2000; Assaad et al., 2022).*

Assumption 3 (Expert-label consistency) *Every alignment label satisfies $y_i = \mathrm{Src}_{\mathcal{G}^\dagger}(V_i)$.*

These assumptions make the feedback well defined but do not make the RCC set unique. Violations of Assumptions 1 or 2 make the APG a task-effective summary rather than the unique data-generating graph. Let $\widetilde{\mathcal{D}}_L = \{(V_i, \widetilde{y}_i)\}_{i=1}^m$ denote a corrupted version of $\mathcal{D}_L = \{(V_i, y_i)\}_{i=1}^m$, where $\widetilde{y}_i \neq y_i$ for at most q incidents. For any fixed graph $\mathcal{G}$, the resulting change in empirical disagreement satisfies $\left|\mathcal{L}_{\mathrm{RCA}}(\mathcal{G}, \widetilde{\mathcal{D}}_L) - \mathcal{L}_{\mathrm{RCA}}(\mathcal{G}, \mathcal{D}_L)\right| \leq q/m$.

Each proposal considers a batch of incidents, and candidate graphs are scored on the full alignment set. A few isolated errors therefore have bounded direct influence, although systematic errors can still alter the proposal path and selected graph. EvoCause consequently targets an RCA-effective graph with low empirical disagreement rather than unique DAG recovery.

3.2 Identifiability under Root-Cause Feedback

Why root labels do not generally identify one DAG. For every labeled root $r \in y_i$, consistency requires $\mathrm{Pa}_{\mathcal{G}}(r) \cap V_i = \varnothing$. For every labeled non-root $v \in V_i \setminus y_i$, however, it only requires $\mathrm{Pa}_{\mathcal{G}}(v) \cap V_i \neq \varnothing$. The latter is a disjunctive constraint: at least one observed parent must exist, but the label does not identify which parent is responsible. Root-cause feedback therefore generally leaves multiple admissible DAGs.

Proposition 1 (Monotone refinement). Let $\mathcal{C}_m$ denote the RCC set induced by the first m labeled incidents. Each additional label imposes a source-set constraint, giving $\mathcal{C}_{m+1} = \mathcal{C}_m \cap \{\mathcal{G} : \mathrm{Src}_{\mathcal{G}}(V_{m+1}) = y_{m+1}\} \subseteq \mathcal{C}_m$. Thus, additional consistent feedback can only shrink or preserve the candidate graph set.

Proposition 2 (Training-set RCA invariance). All graphs in $\mathcal{C}_{\mathrm{RCC}}(\mathcal{D}_L)$ return the same expert root set on every incident in $\mathcal{D}_L$, although they may disagree on unseen incidents and may contain different edge sets. Consequently, training-label consistency alone does not determine which admissible graph will generalize best.

A sufficient condition for uniqueness. Suppose that the true APG is a DAG, its undirected skeleton is known, expert labels are noiseless, and *root-separating coverage* holds: for every adjacency $\{u, v\}$ in the skeleton, the labeled set contains an edge-isolating incident whose relevant alarm-type set is $V_i = \{u, v\}$. For such an incident, the root set is $\{u\}$ if and only if the edge is $u \to v$, and it is $\{v\}$ if and only if the edge is $v \to u$. Every skeleton edge is therefore uniquely oriented, so the RCC set restricted to the known skeleton is a singleton. This condition is sufficient but intentionally strong and need not hold in TeleRCA. Without it, EvoCause targets task-level identification by selecting an RCA-effective representative rather than claiming recovery of the unique data-generating DAG.

3.3 OVERVIEW OF EVOCAUSE

EvoCause comprises a *learning stage* and an *inference stage* (Figure 1). In the learning stage, EvoCause first builds an initial alarm propagation graph $\mathcal{G}_0$ from historical alarm sequences using off-the-shelf causal discovery, and then refines it by aligning it with the downstream RCA task through LLM-based label feedback. In the inference stage, a new incident is resolved by running a transparent graph-based procedure on the refined graph $\mathcal{G}^*$.

3.4 LEARNING STAGE

The learning stage has two components, namely (1) constructing an initial alarm propagation graph from historical sequences, and (2) aligning it with the RCA task through LLM-based label feedback.

Stage I: Initial APG construction. The first stage constructs an initial graph from the label-free historical alarm sequences $\mathcal{D}_{\text{hist}}$:

$$\mathcal{G}_0 = \textsc{CausalDiscovery}(\mathcal{D}_{\text{hist}}). \tag{6}$$

Here $\textsc{CausalDiscovery}(\cdot)$ is any algorithm that returns a DAG over alarm types. We instantiate it with PC (Spirtes et al., 2000), NOTEARS (Zheng et al., 2018), and THPs (Cai et al., 2024), which have different inductive biases. EvoCause treats discovery as a black box. At each refinement round, a deterministic topological order $O(\mathcal{G}) = \textsc{TopologicalOrder}(\mathcal{G})$ is derived from the current DAG solely to give the LLM a compact linearization of causal flow. It is not a second model state, and root-cause prediction depends only on zero in-degree in the incident-induced subgraph.

Stage II: APG alignment via label feedback. The second stage refines $\mathcal{G}_0$ into $\mathcal{G}^*$ using labeled alignment cases $\mathcal{D}_{\text{align}} = \{(\mathbf{s}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{s}_i$ is an incident sequence and $y_i \subseteq V_i$ is its complete expert root set. Following the objective in equation 3, EvoCause searches for a graph that maximizes Case EM on the alignment set.

To optimize equation 3, we adopt a batch-based iterative optimization procedure over N_e epochs. At the start of epoch e, $\mathcal{D}_{\text{align}}$ is randomly permuted to obtain $\mathcal{D}_{\text{align}}^{(e)}$ and then partitioned into batches of size N_b. Each batch triggers one refinement round with four steps.

Step 1: Batch processing. A batch $\mathbf{B}_k$ of N_b labeled cases is drawn from the current epoch's partition. Batching amortizes the cost of LLM invocation and allows each refinement round to aggregate feedback across multiple incidents, reducing sensitivity to any single case.

Step 2: Predict with the current graph. For every case $(\mathbf{s}_i, y_i) \in \mathbf{B}_k$ we run a graph-based RCA procedure on the current graph,

$$\hat{y}_i = \text{RCAEngine}(\mathbf{s}_i, \mathcal{G}), \tag{7}$$

which restricts $\mathcal{G}$ to the alarm types observed in $\mathbf{s}_i$ and returns all zero-in-degree nodes of the incident-induced subgraph. The complete set of such source alarm types is predicted, and the ground-truth number of roots is never provided to the procedure. This yields a per-case triple $(\mathbf{s}_i, \hat{y}_i, y_i)$ whose mismatches are the discrepancies that the next step must explain.

Step 3: LLM proposes graph edits. We collect the incident triples into a batch evidence set $\mathbf{E} = \{(\mathbf{s}_i, \hat{y}_i, y_i)\}_{i \in \mathbf{B}_k}$ and provide it to an LLM $\mathcal{M}$ through the update prompt p_u, together with the current graph $\mathcal{G}$, its derived topological order $O(\mathcal{G})$, and the optimization history $\mathbf{H}$. The LLM proposes only targeted graph edits

$$\Delta\mathcal{G} \;=\; \mathcal{M}(\mathcal{G}, O(\mathcal{G}), \mathbf{E}, \mathbf{H}, p_u)\,, \tag{8}$$

where $\Delta\mathcal{G}$ is a set of directed-edge edits using the operations $\{\textit{add}, \textit{remove}, \textit{reverse}\}$. The LLM response contains only graph updates. The derived topological order $O(\mathcal{G})$ can neither be edited nor returned.

The LLM guides the search for graph updates. A mismatch between predicted and expert source sets can be corrected by several edits, each with different effects on incidents sharing alarm types. Considering all individual directed edge additions, removals, and reversals requires a quadratic number of candidates in the number of alarm types, with combinations expanding the search further. EvoCause narrows this search by using an LLM to interpret source set mismatches, the current graph,

its derived topological order, and optimization history. Deterministic code checks structural validity, while labeled data determine empirical performance. The framework supports arbitrary node identifiers, with meaningful alarm titles providing additional guidance when available.

The optimization history retains the last $K_{\mathcal{H}}$ rounds,

$$\mathbf{H} = \big\{(\mathcal{G}_{o-1}, S_{o-1}), \ldots, (\mathcal{G}_{o-K_{\mathcal{H}}}, S_{o-K_{\mathcal{H}}})\big\}. \tag{9}$$

where $S_{o-1} = (L_{o-1}, F_{o-1})$ is the lexicographic score of graph $\mathcal{G}_{o-1}$ on the full alignment set $\mathcal{D}_{\text{align}}$. Here, L_{o-1} is Case EM and F_{o-1} is Node F1, which is used only to break ties. Specifically,

$$L_{o-1} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\{\text{RCAENGINE}(\mathbf{s}_i, \mathcal{G}_{o-1}) = y_i\}\,. \tag{10}$$

Thus, model selection first maximizes exact root-set recovery over the full alignment set and uses partial node-level recovery only when two candidate graphs have equal Case EM.

Step 4: Validate and apply edits. The parser first checks that every operation is in $\{\textit{add}, \textit{remove}, \textit{reverse}\}$ and that both endpoints belong to $\mathbf{A}$. Valid edits are applied to obtain a tentative graph

$$\mathcal{G}' = \text{APPLYGRAPHUPDATES}(\mathcal{G}, \Delta\mathcal{G}). \tag{11}$$

The tentative graph is accepted if and only if $\mathcal{G}'$ remains a DAG. Otherwise, the complete proposed update is rejected and the current graph is retained. Any topological order needed by the next prompt is then recomputed deterministically from the accepted graph. The refinement loop may visit several structurally valid candidate graphs. After each round, EvoCause evaluates the current graph on the labeled alignment set and retains the graph with the highest Case EM, using Node F1 to break ties.

Termination. The four steps repeat until N_e epochs or N_r rounds are completed, or the graph remains unchanged for c_{stop} consecutive batches. The graph with the highest alignment Case EM is returned as $\mathcal{G}^*$ for downstream RCA. Appendix A.1 summarizes the procedure.

3.5 INFERENCE STAGE

At test time, a new incident $\mathbf{s}_{\text{new}} \in \mathcal{S}_{\text{on}}$ is resolved using the refined graph $\mathcal{G}^*$ alone. The procedure identifies the alarm types present in $\mathbf{s}_{\text{new}}$, restricts $\mathcal{G}^*$ to the subgraph they induce, and returns the complete set of nodes whose in-degree in this induced subgraph is zero. These source alarm types constitute the predicted root-cause set. Formally,

$$\hat{y}_{\text{new}} = \text{RCAENGINE}(\mathbf{s}_{\text{new}}, \mathcal{G}^*). \tag{12}$$

This is the same procedure used to score candidate graphs during learning, so the criterion optimized offline is exactly the one applied online. No LLM call is made at inference time. Every prediction therefore traces back to specific edges in $\mathcal{G}^*$, and every such edge is attributable either to the initial discovery stage or to a recorded LLM graph edit during refinement, providing an audit trail that operators can inspect. Expert labels are used only during offline alignment to score candidate graphs and construct mismatch evidence. Their effect is retained in $\mathcal{G}^*$, so held-out incidents require neither labels nor LLM calls.

4 EXPERIMENTS

We evaluate EvoCause on synthetic networks with known graphs and on TeleRCA, a production telecommunication dataset with expert labels. Synthetic experiments assess RCA, graph reconstruction, and backbone sensitivity using numeric node identifiers. TeleRCA experiments assess RCA, sensitivity to graph initialization, and the value of alarm titles.

4.1 EXPERIMENTAL SETUP

Individual incidents occur relatively independently, while alarm events within each incident follow specific triggering patterns. We therefore randomly divide each dataset into disjoint 80% training and 20% test sets at the incident level. Initial graphs are learned from training data, with 10% of training incidents used for labeled alignment and graph selection. The selected graph is evaluated once on the test set.

Table 1: This table shows results on ten transitively reduced Erdős–Rényi DAGs (mean ± standard deviation). 32B, 4B-Inst., and 4B-Think. denote Qwen3-32B, Qwen3-4B-Instruct-2507, and Qwen3-4B-Thinking-2507, respectively. See Appendices A.3 and A.4 for component and backbone analyses.

Method	**Node F1 (%)** ↑	**Case EM (%)** ↑	**Graph F1 (%)** ↑	**nSHD** ↓
Random	47.23 ± 8.42	6.05 ± 3.18	5.71 ± 4.89	1.8714 ± 0.1421
PC	52.03 ± 6.17	17.80 ± 4.48	64.80 ± 21.58	0.7859 ± 0.7483
NOTEARS	30.03 ± 2.95	7.43 ± 2.38	34.38 ± 11.29	0.8593 ± 0.1554
THPs	37.05 ± 6.54	10.60 ± 3.48	47.93 ± 4.12	0.6882 ± 0.0317
Chain-of-Event (Yao et al., 2024)	53.45 ± 8.11	22.98 ± 6.29	53.71 ± 4.92	1.4795 ± 0.2141
CCCM (Zhang et al., 2024)	58.10 ± 7.11	25.43 ± 5.51	46.23 ± 6.98	1.6179 ± 0.3218
APGNN (Jiang & Bai, 2023)	61.13 ± 7.60	24.17 ± 6.25	51.99 ± 3.94	1.5741 ± 0.1745
RUN (Lin et al., 2024)	60.80 ± 4.69	23.30 ± 4.30	54.67 ± 8.55	1.2931 ± 0.2995
EvoCause-PC (4B-Inst.)	57.78 ± 6.03	21.92 ± 5.41	66.61 ± 14.86	0.6887 ± 0.5529
EvoCause-PC (4B-Think.)	62.80 ± 6.72	26.62 ± 6.98	68.07 ± 10.63	0.5673 ± 0.2100
EvoCause-PC (32B)	$\mathbf{63.62 \pm 6.61}$	$\mathbf{27.20 \pm 6.51}$	$\mathbf{69.39 \pm 8.76}$	$\mathbf{0.5480 \pm 0.1462}$
EvoCause-NOTEARS (32B)	55.09 ± 6.40	20.95 ± 4.68	37.94 ± 8.57	0.7837 ± 0.0645
EvoCause-THPs (32B)	55.74 ± 7.24	20.75 ± 4.41	50.32 ± 3.20	0.6645 ± 0.0283

Evaluation metrics. We evaluate RCA using Node F1 and Case EM (Exact Match). For N test incidents with predicted and true root sets $\widehat{y}_i$ and y_i, Node F1 measures root recovery through micro averaging as $\text{Node F1} = 2\sum_{i=1}^{N} |\widehat{y}_i \cap y_i| / \sum_{i=1}^{N} (|\widehat{y}_i| + |y_i|)$, while $\text{Case EM} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[\widehat{y}_i = y_i]$ measures the proportion of incidents with exactly matching root sets. On synthetic data, we also measure directed edge recovery using $\text{Graph F1} = 2|\widehat{E} \cap E| / (|\widehat{E}| + |E|)$ and normalized structural error using $\text{nSHD} = \text{SHD}(\widehat{G}, G)/|E|$, where $\widehat{E}$ and E are the estimated and true directed edge sets. $\text{SHD}(\cdot)$ calculates the minimum number of edge additions, deletions, and reversals needed to transform $\widehat{G}$ into G, each with unit cost. As TeleRCA has no ground-truth propagation graph, graph-reconstruction metrics are reported only on synthetic data.

Baselines and controlled variants. Across both datasets, we compare unrefined graphs learned by PC, NOTEARS, and THPs with Chain-of-Event (Yao et al., 2024), CCCM (Zhang et al., 2024), APGNN (Jiang & Bai, 2023), and RUN (Lin et al., 2024), using the released default settings for the latter four methods. For PC and NOTEARS, we follow the alarm-transaction representation of Zhang et al. (2021). Random uses a DAG with matched node and edge counts on synthetic data and a random alarm ordering on TeleRCA. EvoCause uses Qwen3-32B with alignment fraction $\rho = 0.1$, batch size $N_b = 20$, epoch count $N_e = 10$, round budget $N_r = 500$, history window $K_{\mathcal{H}} = 5$, and retry budget $N_{\text{retry}} = 3$ per batch. Hyperparameter robustness and label-feedback efficiency are examined in Appendices A.5 and A.6, respectively. On synthetic data, two Qwen3-4B variants assess backbone sensitivity only (Yang et al., 2025). On TeleRCA, EvoCause-PC (Anonymous IDs) replaces alarm titles with fixed anonymous identifiers while keeping all other settings unchanged to test the contribution of human-readable titles.

4.2 Synthetic Data

Dataset. Synthetic data allow evaluation against fully known graphs. We generate ten independent Erdős–Rényi DAGs, each with 50 alarm-type nodes and 70 directed edges before transitive reduction. The reduction preserves all nodes but removes shortcut edges. We retain 2,000 valid incidents per graph, with one to five root alarms per incident. Each active edge generates one offspring event after an exponential delay whose edge-specific mean is sampled once from $[30, 300]$ seconds and fixed across incidents. We report the mean and standard deviation across the ten graphs. Because alarm types use numeric identifiers, these experiments evaluate structural refinement without alarm-name semantics.

Analysis. Table 1 shows that label-feedback refinement improves RCA for all three initial discovery methods, so the benefit is not tied to the inductive bias of PC. With PC initialization, EvoCause raises Node F1 and Case EM by 11.59 and 9.40 percentage points, increases Graph F1 by 4.59 percentage points, and reduces nSHD by 0.2379. RCA improves more strongly than complete

edge recovery for NOTEARS and THPs, which is consistent with expert labels directly constraining incident-level source sets but only indirectly constraining the complete edge set. EvoCause-PC gives the strongest joint RCA and graph-reconstruction result, showing that refinement preserves the value of a strong initial graph. Because synthetic nodes use arbitrary numeric identifiers, these improvements also show that EvoCause can exploit source-set mismatches and graph context without relying on lexical information. The backbone comparisons evaluate sensitivity to the choice of LLM and demonstrate that the framework can operate with different LLM backbones.

4.3 Real-World Telecommunication Data

Dataset. TeleRCA contains 10,922 incidents comprising 485,681 alarm events across 119,039 timestamps, 5,621 network resources, and 194 alarm types, collected from a production telecommunication network in Jakarta, Indonesia, between March and July 2025. Potentially sensitive information was anonymized to meet privacy and publication requirements. The incident-file structure and field definitions are provided in Appendix A.2.

Table 2: This table shows RCA performance on TeleRCA (mean $\pm$ standard deviation over five runs where available). All EvoCause variants use Qwen3-32B. Anonymous IDs replaces alarm titles in the LLM input with fixed anonymous identifiers. Deterministic discovery baselines use single runs.

Method	**Node F1 (%)** $\uparrow$	**Case EM (%)** $\uparrow$
Random	71.69 ± 2.07	62.15 ± 2.79
PC	65.18	53.98
NOTEARS	70.69	61.03
THPs	76.59	68.86
Chain-of-Event	65.82	54.81
CCCM	75.95	67.35
APGNN	84.14 ± 0.91	78.30 ± 1.23
RUN	57.65 ± 0.03	45.03 ± 0.05
EvoCause-PC	$\mathbf{92.58 \pm 0.04}$	$\mathbf{89.61 \pm 0.05}$
EvoCause-PC (Anonymous IDs)	86.46 ± 0.06	81.57 ± 0.08
EvoCause-NOTEARS	92.47 ± 0.07	89.44 ± 0.10
EvoCause-THPs	92.48 ± 0.04	89.46 ± 0.05

Analysis. Table 2 shows that EvoCause achieves similar RCA performance from PC, NOTEARS, and THPs despite their differing initial results, suggesting reduced dependence on discovery initialization. With PC, Node F1 rises from 65.18% to 92.58% and Case EM from 53.98% to 89.61%. Anonymizing alarm titles lowers these scores to 86.46% and 81.57%, indicating that alarm semantics aid graph refinement. Yet the anonymous variant still exceeds the strongest baseline outside EvoCause by 2.32 and 3.27 percentage points, respectively, showing that structural refinement remains effective without lexical information.

5 Conclusion

In this paper, we present EvoCause, a framework that turns historical expert diagnoses into a reusable alarm graph, and introduced TeleRCA, a production benchmark with expert annotations. By combining LLM graph proposals with deterministic validation and selection using labeled incidents, EvoCause improves root cause analysis across different discovery initializations and RCA baselines, while also improving graph reconstruction on synthetic data. The TeleRCA anonymization study further supports the value of alarm titles in guiding refinement. These findings show that expert feedback can improve a shared graph whose predictions require neither labels nor LLM calls at inference. However, recovering a unique causal graph remains an open challenge. Future work will incorporate interventions, temporal evidence, and expert edge constraints to strengthen identifiability, extend the framework to propagation that varies across resources and time, and address latent common causes and noisy labels through richer modeling and active incident selection.

AI USE STATEMENT

In this work, we used generative AI tools for data preprocessing, checking potential issues, and code cleaning. We have not used generative AI tools for other tasks requiring disclosure, and the remaining required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for text polishing. We have reviewed all AI-assisted work, including the revised text, preprocessing procedures, suggested issues, and code changes. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Toufique Ahmed, Supriyo Ghosh, Chetan Bansal, Thomas Zimmermann, Xuchao Zhang, and Saravan Rajmohan. Recommending root-cause and mitigation steps for cloud incidents using large language models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pp. 1737–1749, 2023. doi: 10.1109/ICSE48619.2023.00149.

Charles K. Assaad, Emilie Devijver, and Eric Gaussier. Survey and evaluation of causal discovery methods for time series. *Journal of Artificial Intelligence Research*, 73:767–819, 2022. doi: 10.1613/jair.1.13428.

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 Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 8395–8404. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/assaad23a.html.

Ali Aït-Bachir, Charles K. Assaad, Christophe de Bignicourt, Emilie Devijver, Simon Ferreira, Eric Gaussier, Hosein Mohanna, and Lei Zan. Case studies of causal discovery from it monitoring time series, 2023. URL https://arxiv.org/abs/2307.15678.

Ruichu Cai, Siyu Wu, Jie Qiao, Zhifeng Hao, Keli Zhang, and Xi Zhang. THPs: Topological Hawkes processes for learning causal structure on event sequences. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1):479–493, 2024. doi: 10.1109/TNNLS.2022.3175622.

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 (WWW)*, pp. 2905–2915, 2023. doi: 10.1145/3543507.3583274.

Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, Jun Zeng, Supriyo Ghosh, Xuchao Zhang, Chaoyun Zhang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Tianyin Xu. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems (EuroSys)*, pp. 674–688, 2024. doi: 10.1145/3627703.3629553.

David Maxwell Chickering. Learning equivalence classes of bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

DeepSeek-AI, Daya Guo, Dejian Yang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Zhuangyan Fang, Ruiqi Zhao, Yue Liu, and Yangbo He. On the representation of pairwise causal background knowledge and its applications in causal inference. *Journal of Machine Learning Research*, 26(229):1–73, 2025. URL https://www.jmlr.org/papers/v26/23-0624.html.

F. Richard Guo and Emilija Perković. Efficient least squares for estimating total effects under linearity and causal sufficiency. *Journal of Machine Learning Research*, 23(104):1–41, 2022. URL https://jmlr.org/papers/v23/21-023.html.

Alain Hauser and Peter Bühlmann. Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 13(79): 2409–2464, 2012. URL https://jmlr.org/papers/v13/hauser12a.html.

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

Wenhao Jiang and Yuebin Bai. Apgnn: Alarm propagation graph neural network for fault detection and alarm root cause analysis. *Computer Networks*, 220:109485, 2023. doi: 10.1016/j.comnet.2022.109485.

Ying Li, Yu Kong, Shiwei Yin, and Jianbo Li. Topological-temporal convolution transformer Hawkes process for causal structural learning in telecom networks. *IEEE Internet of Things Journal*, 12(17):36019–36033, 2025. doi: 10.1109/JIOT.2025.3582307.

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, pp. 206–213, 2024. doi: 10.1609/aaai.v38i1.27772.

Matt O'Brien and Frank Bajak. Why did amazon web services crash? here's what it means. https://globalnews.ca/news/8434673/why-amazon-web-services-crash/, 2021. Accessed: 2024-07-25.

Changhua Pei, Zexin Wang, Fengrui Liu, Zeyan Li, Yang Liu, Xiao He, Rong Kang, Tieying Zhang, Jianjun Chen, Jianhui Li, Gaogang Xie, and Dan Pei. Flow-of-Action: SOP-enhanced LLM-based multi-agent system for root cause analysis. In *Companion Proceedings of the ACM Web Conference (WWW)*, 2025. doi: 10.1145/3701716.3715225.

Emilija Perković. Identifying causal effects in maximally oriented partially directed acyclic graphs. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence*, volume 124, pp. 530–539. PMLR, 2020.

Devjeet Roy, Xuchao Zhang, Rashi Bhave, Chetan Bansal, Pedro Henrique B. Las-Casas, Rodrigo Fonseca, and Saravan Rajmohan. Exploring LLM-based agents for root cause analysis. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE Companion)*, pp. 208–219, 2024. doi: 10.1145/3663529.3663841.

Peter Spirtes, Clark N. Glymour, and Richard Scheines. *Causation, Prediction, and Search*. MIT Press, 2nd edition, 2000.

Hanzhang Wang, Zhengkai Wu, Huai Jiang, Yichao Huang, Jiamu Wang, Selcuk Köprü, and Tao Xie. Groot: An event-graph-based approach for root cause analysis in industrial settings. In *Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 419–429, 2021. doi: 10.1109/ASE51524.2021.9678708.

Lu Wang, Chaoyun Zhang, Ruomeng Ding, Yong Xu, Qihang Chen, Wentao Zou, Qingjun Chen, Meng Zhang, Xuedong Gao, Hao Fan, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. Root cause analysis for microservice systems via hierarchical reinforcement learning from human feedback. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 5116–5125, 2023. doi: 10.1145/3580305.3599934.

Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Jihong Wang, Fengbin Yin, Lunting Fan, Lingfei Wu, and Qingsong Wen. RCAgent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 4966–4974, 2024. doi: 10.1145/3627673.3680016.

Zhiqiang Xie, Yujia Zheng, Lizi Ottens, Kun Zhang, Christos Kozyrakis, and Jonathan Mace. Cloud Atlas: Efficient fault localization for cloud systems using language models and causal insight. *arXiv preprint arXiv:2407.08694*, 2024. URL https://arxiv.org/abs/2407.08694.

Junjielong Xu, Qinan Zhang, Zhiqing Zhong, Shilin He, Chaoyun Zhang, Qingwei Lin, Dan Pei, Pinjia He, Dongmei Zhang, and Qi Zhang. Openrca: Can large language models locate the root cause of software failures? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d29b8d53678015079e1d245c023e49d2-Abstract-Conference.html.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Zhenhe Yao, Changhua Pei, Wenxiao Chen, Hanzhang Wang, Liangfei Su, Huai Jiang, Zhe Xie, Xiaohui Nie, and Dan Pei. Chain-of-Event: Interpretable root cause analysis for microservices through automatically learning weighted event causal graph. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE Companion)*, pp. 50–61, 2024. doi: 10.1145/3663529.3663827.

Lei Zan, Charles K. Assaad, Emilie Devijver, Eric Gaussier, and Ali Aït-Bachir. On the fly detection of root causes from observed data with application to IT systems. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 5062–5069. ACM, 2024. doi: 10.1145/3627673.3680010.

Lei Zan, Keli Zhang, Ruichu Cai, and Lujia Pan. CAMA: Enhancing mathematical reasoning in large language models with causal knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 34566–34574, 2026. doi: 10.1609/aaai.v40i41.40756. URL https://ojs.aaai.org/index.php/AAAI/article/view/40756.

Ke Zhang, Xu Zhu, Jiale Zheng, Jie Cao, Keli Zhang, and Lujia Pan. Cluster-aware causal discovery framework for root cause analysis of base station alarms. In *2024 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6. IEEE, 2024. doi: 10.1109/GCWkshp64532.2024.11101284.

Keli Zhang, Marcus Kalander, Min Zhou, Xi Zhang, and Junjian Ye. An influence-based approach for root cause alarm discovery in telecom networks. In *Service-Oriented Computing – ICSOC 2020 Workshops*, pp. 124–136. Springer, 2021. doi: 10.1007/978-3-030-76352-7_16. URL https://arxiv.org/abs/2105.03092.

Nengwen Zhao, Junjie Chen, Xiao Peng, Honglin Wang, Xinya Wu, Yuanzong Zhang, Zikai Chen, Xiangzhong Zheng, Xiaohui Nie, Gang Wang, et al. Understanding and handling alert storm for online service systems. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice*, pp. 162–171, 2020.

Xun Zheng, Bryon Aragam, Pradeep K. Ravikumar, and Eric P. Xing. Dags with no tears: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, volume 31, pp. 9492–9503, 2018.