An Influence-based Approach for Root Cause Alarm Discovery in Telecom Networks

Keli Zhang*, Marcus Kalander*, Min Zhou, Xi Zhang and Junjian Ye

Noah's Ark Lab, Huawei Technologies

Abstract. Alarm root cause analysis is a significant component in the day-to-day telecommunication network maintenance, and it is critical for efficient and accurate fault localization and failure recovery. In practice, accurate and self-adjustable alarm root cause analysis is a great challenge due to network complexity and vast amounts of alarms. A popular approach for failure root cause identification is to construct a graph with approximate edges, commonly based on either event co-occurrences or conditional independence tests. However, considerable expert knowledge is typically required for edge pruning. We propose a novel data-driven framework for root cause alarm localization, combining both causal inference and network embedding techniques. In this framework, we design a hybrid causal graph learning method (HPCI), which combines Hawkes Process with Conditional Independence tests, as well as propose a novel Causal Propagation-Based Embedding algorithm (CPBE) to infer edge weights. We subsequently discover root cause alarms in a real-time data stream by applying an influence maximization algorithm on the weighted graph. We evaluate our method on artificial data and real-world telecom data, showing a significant improvement over the best baselines.

Keywords: Network Management · Root Cause Analysis · Alarm Correlation Analysis · Influence Maximization.

1 Introduction

Recent years have seen rapid development in cellular networks, both in increasing network scale and complexity coupled with increasing network performance demands. This growth has made the quality of network management an even greater challenge and puts limits on the analysis methods that can be applied. In cellular networks, anomalies are commonly identified through alarms. A large-scale network can generate millions of alarms during a single day. Due to the interrelated network structure, a single fault can trigger a flood of alarms from multiple devices. Traditionally, to recover after a failure, an operator will analyze all relevant alarms and network information. This can be a slow and time-consuming process. However, not all alarms are relevant. There exists a subset of alarms that are the most significant for fault localization. We denote these as root cause alarms, and our main goal is to intelligently identify these alarms.

[*] These authors contributed equally to this work.

There exist abundant prior research in the areas of Root Cause Analysis (RCA) and fault localization. However, most proposed methods are highly specialized and take advantage of specific properties of the deployed network, either by using integrated domain knowledge or through particular design decisions [2–4]. A more general approach is to infer everything from the data itself.

In our proposed alarm RCA system, we create an influence graph to model alarm relations. Causal inference is used to infer an initial causal graph, and then we apply a novel Causal Propagation-Based Embedding (CPBE) algorithm to supplement the graph with meaningful edge weights. To identify the root cause alarms, we build upon ideas in how influence propagates in social networks and view the problem as an influence maximization problem [10], i.e., we want to discover the alarms with the largest influence. When a failure transpires, our system can automatically perform RCA based on the sub-graph containing the involved alarms and output the top-K most probable root cause alarms.

In summary, our main contributions are as follows:

– We design a novel unsupervised approach for root cause alarm localization that integrates casual inference and influence maximization analysis, making the framework robust to causal analysis uncertainty without requiring labels.
– We propose HPCI, a Hawkes Process-based Conditional Independence test procedure for causal inference.
– We further propose CPBE, a Causal Propagation-Based Embedding algorithm based on network embedding techniques and vector similarity to infer edge weights in causality graphs.
– Extensive experiments on a synthetic and a real-world citywide dataset show the advantages and usefulness of our proposed methods.

2 Related work

Root cause alarms. There are various ways to discover alarm correlations and root cause alarms. Rules and experience of previous incidents are frequently used. In more data-driven approaches, pattern mining techniques that compress alarm data can assist in locating and diagnosing faults [24]. Abele et al. [1] propose to find root cause alarms by combining knowledge modeling and Bayesian networks. To use an alarm clustering algorithm that considers the network topology and then mine association rules to find root cause alarms was proposed in [21].

Graph-based root cause analysis. Some previous works depend on system dependency graphs, e.g., Sherlock [2]. A disadvantage is the requirement of exact conditional probabilities, which is impractical to obtain in large networks. Other systems are based on causality graphs. G-RCA [4] is a diagnosis system, but its causality graph is configured by hand, which is unfeasible in large scale, dynamic environments. The PC algorithm [19] is used by both CauseInfer [3] and [11] to estimate DAGs, which are then used to infer root causes. However, such graphs can be very unreliable. Co-occurrence and Bayesian decision theory are used in [13] to estimate causal relations, but it is mainly based on log event heuristics and is hard to generalize. Nie et al. [16] use FP-Growth and lag correlation to build a causality graph with edge weights added with expert feedback.

3 Preliminaries

In this section, we shortly review the two key concepts that our proposed method depends upon, Hawkes process [7] and the PC algorithm [20].

Hawkes Process. This is a popular method to model continuous-time event sequences where past events can excite the probability of future events. The keystone of Hawkes process is the conditional intensity function, which indicates the occurrence rate of future events conditioned on past events, denoted by $\lambda_d(t)$, where $u \in \mathcal{C} = \{1, 2, ..., U\}$ is an event type. Formally, given an infinitely small time window $[t, t + \varDelta t)$, the probability of a type-u event occurring in this window is $\lambda_d(t)\varDelta t$. For U-dimensional Hawkes process with event type set $\mathcal{C}$, each dimension u has a specific form of conditional intensity function defined as

$$\lambda_u(t) = \mu_u + \sum_{v \in \mathcal{C}} \sum_{t_i < t} k_{uv}(t - t_i), \tag{1}$$

where $\mu_u \geq 0$ is the background intensity for type-u events and $k_{uv}(t) \geq 0$ is a kernel function indicating the influence from past events. An exponential kernel is most frequently used, i.e., $k_{uv}(t) = \alpha_{uv} e^{-\beta_{uv}(t)}$, where α_{uv} captures the degree of influence of type-v events to type-u events and β_{uv} controls the decay rate. The parameters are commonly learned by optimizing a log-likelihood function. Let $\mu = (\mu_u) \in R^U$ be the background intensities, and $A = (\alpha_{uv}) \in R^{U \times U}$ the influence matrix reflecting the certain causality between event types. For a set of event sequences $\mathcal{S} = \{S_1, S_2, ..., S_m\}$, where each event sequence $S_i = \{(a_{ij}, t_{ij})\}_{j=1}^{n_i}$ is observed during a time period of $[0, T_i]$, and each pair $((a_{ij}, t_{ij}))$ represents an event of type a_{ij} that occurred at time t_{ij}. The log-likelihood of a Hawkes process model with parameters $\varTheta = \{\mu, A\}$ can then be expressed as

$$\mathcal{L}(\mu, A) = \sum_{i=1}^{m} (\sum_{j=1}^{n_i} \log \lambda_{a_{ij}}(t_{ij}) - \sum_{u=1}^{U} \int_0^{T_i} \lambda_u(t) dt). \tag{2}$$

The influence matrix A is generally sparse or low-rank in practice, hence, adding penalties into $\mathcal{L}(\mu, A)$ is common. For instance, Zhou [25] used a mix of Lasso and nuclear norms to constrain A to be both low-rank and sparse by using

$$\min_{\mu \geq 0, A \geq 0} -\mathcal{L}(\mu, A) + \rho_1 ||A||_1 + \rho_2 ||A||_*, \tag{3}$$

where $|| \cdot ||_1$ is the L_1-norm, and $|| \cdot ||_* = \sum_{i=1}^{rankA} \sigma_i$ is the nuclear norm. The parameters ρ_1 and ρ_2 controls their weights. A number of algorithms can be applied to solve the above learning problem, more details can be found in [22].

PC Algorithm. This algorithm is frequently used for learning directed acyclic graphs (DAGs) due to its strong causal structure discovery ability [23]. Conditional Independence (CI) tests play a central role in the inference. A significance level p is used as a threshold to determine if an edge should be removed or retained. Formally, given a variable set Z, if X is independent of Y, denoted as $X \perp\!\!\!\perp Y | Z$, the edge between X and Y will be removed, otherwise it will be kept in the causal graph. A rigorous description can be found in [9].

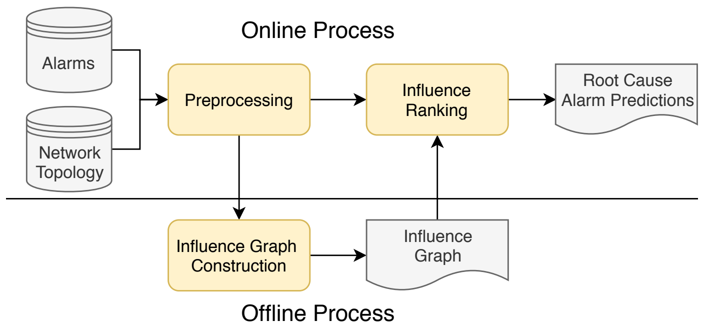

Fig. 1. Architecture of the proposed system.

The G-square test and Fisher-Z test are two common realizations for conditional independence testing in causal inference [18]. The G-square test is used for testing independence of categorical variables using conditional cross-entropy while the Fisher-Z test evaluates conditional independence based on Pearson's correlation. CI tests assume that the input is independent and identically distributed. In our alarm RCA scenario, the size of the time window depends on the network characteristics and needs to be selected to ensure that causal alarms exist in one window and the data between different windows are independent.

4 System Overview

Our proposed framework consists of two main procedures: influence graph creation and alarm ranking. A system overview can be found in Figure 1. The alarm preprocessing module is shared and handles alarm filtering and aggregation with consideration to the network topology.

The influence graph is constructed using historical alarm transactions and is periodically recreated. It is comprised of alarm types as nodes and their inferred relations as edges. To create the graph, we first exploit causal inference methodology to infer an initial alarm causality graph structure by applying HPCI, a hybrid method that merges Hawkes process and conditional independence tests. We further apply a network embedding technique, CPBE, to infer the edge weights. The alarm stream is monitored in real-time. When a failure transpires, the system attempts to discover the underlying root cause alarms. The related alarms are aggregated with the created influence graph and are ranked by their influence to determine the top-K most probable root cause alarm candidates. The alarm candidates are then given to the network operators to assist in handling the network issue.

5 Methodology

This section introduces the key components in our system; alarm preprocessing, the influence graph construction, and how the influence ranking is done. We start by presenting the data and its required preprocessing and aggregation steps.

5.1 Data and Alarm Preprocessing

Network Topology. This is the topological structure of the connections between network devices. Connected network devices will interact with each other. If a failure occurs on one device, then any connected devices can be affected, triggering alarms on multiple network nodes.

Alarms. Network alarms are used to identify faults in the network. Each alarm record contains information about occurrence time, network device where the alarm originated, alarm type, etc. In practice, any alarms with missing key information are useless and removed. Furthermore, alarm types that are either systematic or highly periodical are also removed. These types of alarms are irrelevant for root cause analysis since they will be triggered regardless if a fault occurred or not.

Alarm Preprocessing. We partition the raw alarm data into alarm sequences and alarm transactions in three steps as follows.

1. Devices in connected sub-graphs of the network can interact, i.e., alarms from these devices can potentially be related to the same fault. Consequently, we first aggregate alarms from the same sub-graph together.
2. Alarms related to the same failure will generally occur together within a short time interval. We thus further partition the alarms based on their occurrence times. Alarms that occurred within the same time window w_i are grouped and sorted by time. The window size can be adjusted depending on network characteristics. We define each group as an alarm sequence, denoted as $S_i = \{(a_{ij}, t_{ij})\}_{j=1}^{n_i}$, where w_i is the window, $a_{ij} \in A$ is the alarm type, $t_{ij} \in w_i$ is occurrence time, and n_i the number of alarms.
3. Each alarm sequence is transformed into an alarm transaction denoted by $T_i = \{(a, t, n) | a \in A_i\}$, where a, t, n indicates the alarm type, the earliest occurrence time and the number of occurrences, respectively. Different from S_i, A_i contains a single element for each alarm type in window w_i.

5.2 Alarm Influence Graph Construction

In this section, we elaborate on the construction of the alarm influence graph. The graph has the alarm types as nodes and their relation as the edges. First, an initial causal structure DAG is inferred by a hybrid causal structure learning method (HPCI). Subsequently, edge weights are inferred using a novel network embedding method (CPBE).

HPCI. A multi-dimensional Hawkes process can capture certain causalities behind event types, i.e., the transpose of the influence matrix A can be seen as the adjacency matrix of the causal graph for event types. However, redundant or indirect edges tend to be discovered since the conditional intensity function can not perfectly model real-world data and due to the difficulty in capturing the instantaneous causality.

To reduce this weakness, we propose a hybrid algorithm HPCI that is based on Hawkes process and the PC algorithm. HPCI is used to discover the causal

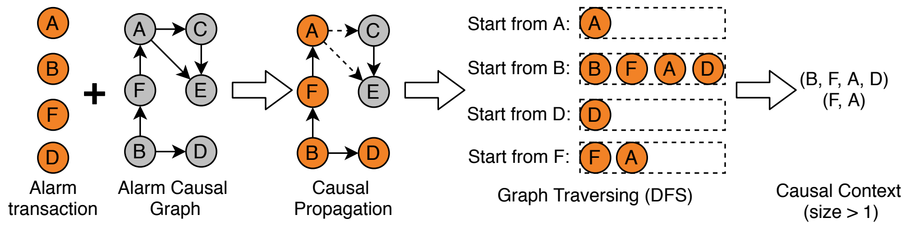

Fig. 2. Context generation procedure for CPBE.

structure for the alarm types in our alarm RCA scenario. The main procedure can be expressed in three steps. (1) Use multi-dimensional Hawkes process without penalty to capture the influence intensities among the alarm types. We use the alarm sequences $\mathcal{S} = \{S_i\}$ as input and obtain an initial weighted graph. The weights on an edge (u, v) is the influence intensity $\alpha_{uv} > 0$, reflecting the expectation of how long it takes for a type-u event to occur after an type-v event. All edges with positive weights are retained. (2) Any redundant and indirect causal edges are removed using CI tests. We use the alarm transactions $\mathcal{T} = \{T_i\}$ as input and for each alarm a_i the sequence of alarm occurrences $N_i = \{n | T_k \in \mathcal{T}, (a, t, n) \in T_k, a = a_i\}$ is extracted. Note that n can be 0 if an alarm type is not present in a window w_i. For each pair of alarm types (a_i, a_j), the CI test of their respective occurrence sequences is used to test for independence and remove edges. The output is a graph with unwanted edges removed. (3) Finally, we iteratively remove the edge with the smallest intensity until the graph is a DAG. Our final causal graph is denoted as G^C.

We select CI tests to enforce sparsity in the causal graph in the second step. Compared to adding penalty terms such as $L1$-norm, the learning procedure is more interpretable, and our experiments show more robust results.

Edge Weights Inference. The causal graph G^C learned by HPCI is a weighted graph, however, the weights do not account for global effects on the causal intensities. Hence, to encode more knowledge into the graph, we propose a novel network embedding-based weight inference method, Causal Propagation-Based Embedding (CPBE). CPBE consists mainly of two steps; (1) For each node u, we obtain a vector representation $Z_u \in \mathcal{R}^L$ using a novel network embedding technique. (2) Use vector similarity to compute edge weights between nodes.

The full CPBE algorithm is shown in Algorithm 1. CPBE uses a new procedure to generate a context for the skip-gram model [15] (lines 1-9). This procedure is also illustrated in Figure 2. In essence, for each historical alarm transaction $T_i \in \mathcal{T}$, we use the learned causality graph G^C and extract a causal propagation graph G_i^{PC}, where only the nodes corresponding to alarm types in T_i are retained. Starting from each node in G_i^{PC}, we traverse the graph to generate a node-specific causal context. During the traversal for a node u, only nodes that have a causal relation with u are considered. There are various possible traversing strategies, e.g., depth-first search (DFS) and RandomWalk [6]. The skip-gram model is applied to the generated contexts to obtain an embedding vector $Z_u \in \mathcal{R}^L$ for each node u. Finally, the edge weight between two nodes is

Algorithm 1 Causal Propagation-Based Embedding (CPBE)

```
Input: Alarm Transactions $\mathcal{T} = \{T_i\}$; Causal Graph $G^C = (V, E)$;
 1: $C = \{\}, E_w = \{\}$;
 2: for $T_i \in T$ do:
 3:     $G_i^{PC}$ = ConstructPropagationGraph($T_i, G^C$)
 4:     for $alarm\_node \in T_i$ do:
 5:         $C_{alarm\_node}$ = GraphTraversing($G_i^{PC}, alarm\_node$)
 6:         $C = C \cup C_{alarm\_node}$
 7:     end for
 8: end for
 9: $Z \leftarrow$ skip-gram($C$) to map nodes to embedding vectors
10: for $(u, v) \in E$ do:
11:     $w = Cosine(Z_u, Z_v)$
12:     $E_w = E_w \cup (u, v, w)$
13: end for
Output: Alarm Influence Graph $G^I = (V, E_w)$;
```

set to be the cosine similarity of their associated vectors. We denote the final weighted graph as the alarm influence graph G^I.

5.3 Root Cause Alarm Influence Ranking

This section describes how the alarm influence graph G^I is applied to an alarm transaction to identify the root cause alarms. For each alarm transaction $T_i \in \mathcal{T}$, an alarm propagation graph G_i^{PI} is created with the relevant nodes $v \in T_i$ and applicable edges $\{(u, v, w) | u, v \in T_i\}$. Any nodes corresponding to alarms not present in T_i are removed. The process is equivalent to how G_i^{PC} is created from the causal graph G^C. The alarms in each propagation sub-graph are then ranked independently. The process is illustrated in Figure 3.

We consider the problem of finding the root cause alarm as an influence maximization problem [10]. We want to discover a small set of K *seed nodes* that maximizes the influence spread under an influence diffusion model. A suitable model is the independent cascade model, which is widely used in social network analysis. Following this model, each node v is activated by each of its neighbors independently based on an influence probability $p_{u,v}$ on each edge (u, v). These probabilities directly correspond to the learned edge weights. Given a seed set S_0 to start with at $t = 0$, at step $t > 0$, $u \in S_{t-1}$ tries to activate its outgoing inactivated neighbors $v \in \mathcal{N}^{out}(u)$ with probability $p_{u,v}$. Activated nodes are added to S_t and the process terminates when $|S_t| = 0$, i.e., when no nodes further nodes are activated. The influence of the seed set S_0 is then the expected number of activated nodes when applying the above stochastic activation procedure.

There are numerous algorithms available to solve the influence maximization problem [12]. In our scenario, each graph G_i^{PI} is relatively small and the actual algorithm is thus less important. We directly select the Influence Ranking Influence Estimation algorithm (IRIE) [8] for this task. IRIE estimates the influence

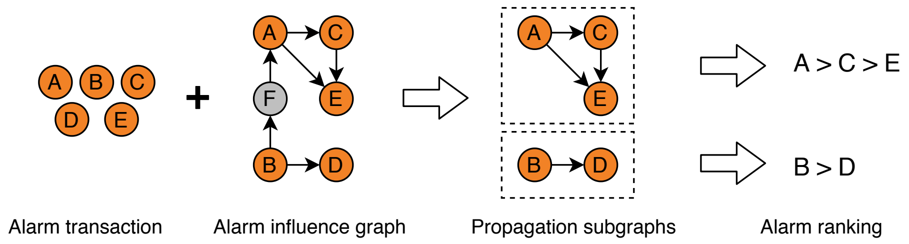

Fig. 3. Processing flow from alarm transaction to ranked alarms.

$r(u)$ for each node u by deriving a system of n linear equations with n variables. The influence of a node u comprises of its own influence, 1, and the sum of the influences it propagates to its neighbors.

6 Evaluation

In this section, we present the experimental setup and evaluation results. We perform two main experiments, one to verify the correctness of our causal graph and a second experiment to evaluate the root cause identification accuracy. The first experiment is performed on both synthetic and real-world data, while the second is completed on the real-world dataset. The datasets and code are available at https://github.com/shaido987/alarm-rca.

Synthetic Data Generation. The synthetic event sequences are generated in four steps. (1) We randomly generate a DAG G with an average out-degree d with N event types. We set d to 1.5 to emulate the sparsity property of our real-world dataset. (2) For each edge (u, v), a weight α_{uv} is assigned by uniform random sampling from a range $r \in [(0.01, 0.05), (0.05, 0.1), (0.1, 0.5), (0.5, 1.0)]$. (3) For each event type $u \in U$, we assign a background intensity μ_u by uniform random sampling from $(0.001, 0.005)$. (4) Following Ogata [17], we use α_{uv} and μ_u as parameters of a Multi-dimensional Hawkes process and simulate event sequences. We generate event sequences of length $T = 14$ days while ensuring that the total number of events is greater than 10,000.

Real-world Dataset. The dataset was collected from a major cellular carrier in a moderate-sized city in China between Aug 4th, 2018 and Oct 24th, 2018. After preprocessing, it consists of 672,639 alarm records from 3,818 devices with 78 different alarm types. Due to the difficulty of labeling causal relations, we only have the ground-truth causal relations for a subset of 15 alarm types, 44 directed edges in the graph. Furthermore, we have also obtained the ground-truth root cause alarms in a random sample of 6,000 alarm transactions. These are used to evaluate the root cause localization accuracy.

6.1 Causal Graph Structure Correctness

We evaluate our proposed HPCI method and the accuracy of the discovered causal graphs. We use four frequently used causal inference methods for sequential data as baselines.

Table 1. F1-scores on the synthetic dataset for different number of event types and α.

α	N	PS-GS	PS-FZ	PCTS	HPADM4	HPCI
(0.01,0.05)	10	0.286	0.174	0.283	**0.566**	0.200
	20	0.133	0.321	0.156	0.306	**0.604**
	30	0.216	0.267	0.104	0.229	**0.357**
(0.05,0.1)	10	0.500	0.529	0.283	0.500	**0.867**
	20	0.367	0.585	0.155	0.323	**0.806**
	30	0.265	0.484	0.227	0.227	**0.756**
(0.1,0.5)	10	0.467	0.811	0.278	0.517	**0.933**
	20	0.621	0.889	0.151	0.306	**0.984**
	30	0.495	0.845	0.103	0.227	**0.967**
(0.5,1.0)	10	0.800	0.722	0.272	0.517	**0.929**
	20	0.708	0.906	0.151	0.302	**0.983**
	30	0.433	0.845	0.103	0.227	**0.967**

- PC-GS: PC algorithm with G-square CI test.
- PC-FZ: PC algorithm with Fisher-Z CI test.
- PCTS: Improved PC algorithm for causal discovery in time series [14].
- HPADM4: Multi-dimensional Hawkes process with exponential parameterization of the kernels and a mix of $L1$ and nuclear-norm [25].

The significance level p in the conditional independence tests included in the methods are all set to 0.05. The size of time window w for aggregating event sequences is set to 300 seconds, the maximum lag $\tau_{max} = 2$ in PCTS, and the penalization level in HPADM4 is set to the default 1,000. Furthermore, the decay parameter β in Hawkes process is set to 0.1, and we select Fisher-Z as the CI test in our HPCI algorithm. For evaluation, we define three metrics as follows.

$$Precision = \frac{|P \cap S|}{|P|}, \quad Recall = \frac{|P \cap S|}{|S|}, \quad F1\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall},$$

where P is the set of all directed edges in the learned causal graph G^C and S is the set of ground-truth edges.

Results. The F1-scores using synthetic data with $N \in [10, 15, 20]$ are shown in Table 1. As shown, HPCI outperforms the baselines for nearly all settings of N and α. However, HPADM4 obtains the best result for $N = 10$ and low α, this is due to the distribution of event occurrence intervals being sparse which makes the causal dependency straightforward to capture using a Hawkes process. However, for higher N or α the events will be denser. Thus, Hawkes process has trouble distinguishing instantaneous causal relations, especially when events co-occur. The use of CI tests in HPCI helps to distinguish these instantaneous causal relations by taking another perspective in which causality is discovered based on distribution changes in the aggregated data without considering the time-lagged information among events. HPCI thus achieves better results. The

Table 2. Results of the causal graph structure evaluation on the real-world dataset.

Method	Precision	Recall	F1-score
PC-GS	0.250	0.159	0.194
PC-FZ	0.452	0.432	0.442
PCTS	0.220	**0.864**	0.350
HPADM4	0.491	0.614	0.545
HPCI	**0.634**	0.591	**0.612**

use of time aggregation is disadvantageous for PCTS due to its focus on time series, which can partly explain its comparatively worse results.

The results on the real-world data are shown in Table 2. HPCI performs significantly better than all baselines in precision and F1-score, while PTCS obtains the highest recall. PTCS also has significantly lower precision, indicating more false positives. PCTS is designed for time series, however, those may be periodic, which can give higher lagged-correlation values leading to more redundant edges. HPCI instead finds a good balance between precision and recall. The competitive result indicates that the causality behind the real alarm data conforms to the assumptions of HPCI to a certain extent.

6.2 Root Cause Alarm Identification

We evaluate the effectiveness of CPBE and the root cause alarm accuracy on the real-world dataset. We use the causal graph structure created by HPCI as the base and augment it with the 44 known causal ground-truths. The causal graph is thus as accurate as possible. CPBE is compared with four baseline methods, all used for determining edge weights.

- IT, directly use the weighted causal graph discovered by HPCI with the learned influence intensities as edge weights.
- Pearson, uses the aligned Pearson correlation of each alarm pair [16].
- CP, the weights of an edge (u, v) is set to $\frac{A_{uv}}{A_u}$ where A_{uv} is the number of times u and v co-occur in a window, and A_u is the total number of u alarms.
- ST, a static model with maximization likelihood estimator [5]. It is similar to CP, but A_{uv} represents the number of times u occurs before v.

For each method, IRIE is used to find the top-K most likely root cause alarms in each of the 6,000 labeled alarm transactions. For IRIE, we use the default parameters. We attempt to use RandomWalk, BFS, and DFS for traversal in CPBE, as well as different Skip-gram configurations with $w \in [1, 5]$ and vector length $L \in [10, 30]$. However, there is no significant difference in the outcome, indicating that CPBE is insensitive to these parameter choices on our data. The results for different K when using RandomWalk are shown in Table 3. As shown, CPBE outperforms the baselines for all K. For $K = 1$, CPBE achieves an accuracy of 61.8% which, considering that no expert knowledge is integrated into the system, is an excellent outcome. Moreover, the running time of CPBE is

Table 3. Root cause alarm identification accuracy using different edge weight inference strategies together with IRIE for alarm ranking at different K.

Method	K=1	K=2	K=3	K=4	K=5
IT	0.576	0.590	0.672	0.810	0.900
Pearson	0.407	0.435	0.456	0.486	0.486
CP	0.474	0.640	0.730	0.790	0.840
ST	0.439	0.642	0.750	0.785	0.814
CPBE	**0.618**	**0.752**	**0.851**	**0.929**	**0.961**

around 10 seconds and IRIE takes 325 seconds for all 6,000 alarm transactions. This is clearly fast enough for system deployment.

7 Conclusion

We present a framework to identify root cause alarms of network faults in large telecom networks without relying on any expert knowledge. We output a clear ranking of the most crucial alarms to assist in locating network faults. To this end, we propose a causal inference method (HPCI) and a novel network embedding-based algorithm (CPBE) for inferring network weights. Combining the two methods, we construct an alarm influence graph from historical alarm data. The learned graph is then applied to identify root cause alarms through a flexible ranking method based on influence maximization. We verify the correctness of the learned graph using known causal relation and show a significant improvement over the best baseline on both synthetic and real-world data. Moreover, we demonstrate that our proposed framework beat the baselines in identifying root cause alarms.

References

1. Abele, L., Anic, M., et al.: Combining knowledge modeling and machine learning for alarm root cause analysis. IFAC Proceedings Volumes **46**(9), 1843–1848 (2013)
2. Bahl, P., Chandra, R., et al.: Towards highly reliable enterprise network services via inference of multi-level dependencies. In: ACM SIGCOMM Computer Communication Review. vol. 37, pp. 13–24. ACM (2007)
3. Chen, P., Qi, Y., et al.: Causeinfer: automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems. In: INFOCOM, 2014 Proceedings IEEE. pp. 1887–1895. IEEE (2014)
4. Ge, Z., Yates, J., et al.: Grca: A generic root cause analysis platform for service quality management in large isp networks. In: ACM ACM Conference on Emerging Networking Experiments and Technologies (2010)
5. Goyal, A., Bonchi, F., et al.: Learning influence probabilities in social networks. In: Proceedings of the third ACM international conference on Web search and data mining. pp. 241–250. ACM (2010)
6. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 855–864. ACM (2016)

7. Hawkes, A.G.: Spectra of some self-exciting and mutually exciting point processes. Biometrika **58**(1), 83–90 (1971)
8. Jung, K., Heo, W., et al.: Irie: Scalable and robust influence maximization in social networks. In: Data Mining (ICDM), 2012 IEEE 12th International Conference on. pp. 918–923. IEEE (2012)
9. Kalisch, M., Bühlmann, P.: Estimating high-dimensional directed acyclic graphs with the PC-algorithm. Journal of Machine Learning Research **8**, 613–636 (2007)
10. Kempe, D., Kleinberg, J., et al.: Maximizing the spread of influence through a social network. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 137–146. ACM (2003)
11. Kobayashi, S., Otomo, K., et al.: Mining causality of network events in log data. IEEE Transactions on Network and Service Management **15**(1), 53–67 (2018)
12. Li, Y., Fan, J., et al.: Influence maximization on social graphs: A survey. IEEE Transactions on Knowledge and Data Engineering (2018)
13. Lou, J.G., Fu, Q., et al.: Mining dependency in distributed systems through unstructured logs analysis. SIGOPS Operating Systems Review **44**(1), 91–96 (2010)
14. Meng, Y., Zhang, S., Sun, Y., Zhang, R., Hu, Z., Zhang, Y., Jia, C., Wang, Z., Pei, D.: Localizing failure root causes in a microservice through causality inference. In: 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS). pp. 1–10. IEEE (2020)
15. Mikolov, T., Sutskever, I., et al.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
16. Nie, X., Zhao, Y., et al.: Mining causality graph for automatic web-based service diagnosis. In: Performance Computing and Communications Conference (IPCCC), 2016 IEEE 35th International. pp. 1–8 (2016)
17. Ogata, Y.: On lewis' simulation method for point processes. IEEE transactions on information theory **27**(1), 23–31 (1981)
18. Peters, J., Mooij, J.M., et al.: Causal discovery with continuous additive noise models. The Journal of Machine Learning Research **15**(1), 2009–2053 (2014)
19. Spirtes, P., Glymour, C.: An algorithm for fast recovery of sparse causal graphs. Social science computer review **9**(1), 62–72 (1991)
20. Spirtes, P., Glymour, C.N., et al.: Causation, prediction, and search. MIT press (2000)
21. Su, C., Hailong, Z., et al.: Association mining analysis of alarm root-causes in power system with topological constraints. In: Proceedings of the 2017 International Conference on Information Technology. pp. 461–468. ACM (2017)
22. Veen, A., Schoenberg, F.P.: Estimation of space–time branching process models in seismology using an em–type algorithm. Journal of the American Statistical Association **103**(482), 614–624 (2008)
23. Wang, P., Xu, J., et al.: Cloudranger: Root cause identification for cloud native systems. In: 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). pp. 492–502. IEEE (2018)
24. Zhang, X., Bai, Y., et al.: Network alarm flood pattern mining algorithm based on multi-dimensional association. In: Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems. pp. 71–78. ACM (2018)
25. Zhou, K., Zha, H., et al.: Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In: Artificial Intelligence and Statistics. pp. 641–649 (2013)