Journal of Computer Science and Electrical Engineering
Print ISSN: 2663-1938
Online ISSN: 2663-1946
DOI: https://doi.org/10.61784/jcsee4158

MCG: MARGINAL–CONDITIONAL GRAPH LEARNING FOR JOINT ANOMALY DETECTION AND ROOT CAUSE LOCALIZATION IN DISTRIBUTED CLOUD SYSTEMS

SuChuan Xing[1], YiHan Wang[2*]
[1]*Duke University, Durham 27708, NC, USA.*
[2]*University of Pennsylvania, Philadelphia 19104, PA, USA.*
[*]*Corresponding Author: YiHan Wang*

Abstract: Distributed cloud systems expose large numbers of interdependent performance metrics, and operators must both detect anomalies and identify the component that caused them. Deep multivariate detectors provide per-metric anomaly evidence, yet this evidence mixes the faulty component with the many metrics that merely inherit its effects, whereas most root cause analysis (RCA) techniques require an accurate call topology or costly causal discovery at incident time. This paper proposes MCG, a Marginal–Conditional Graph network that unifies anomaly detection and topology-free root cause localization. MCG learns a sparse directed dependency graph among metrics and couples two predictive heads: a marginal head that forecasts each metric from history only, and a conditional head that additionally propagates the innovations, i.e., the unexpected deviations, of the learned parents of each metric at the current time step. The marginal residual drives detection and metric-level diagnosis, while the conditional residual isolates the part of a deviation that upstream metrics cannot explain, which is the statistical signature of a fault intervention. Experiments on the public SMD server-monitoring dataset and on a simulated microservice benchmark with 120 injected faults compare MCG with seven detectors and five RCA methods. MCG attains the fourth-best average detection rank across the two evaluated datasets, competitive dimension-level diagnosis accuracy on SMD (HitRate@150% of 0.587), and an AC@1 of 0.733 in root-cause service localization without any topology knowledge, the best result among topology-free methods.
Keywords: AIOps; Anomaly detection; Root cause analysis; Graph neural networks; Multivariate time series; Microservices; Cloud computing

1 INTRODUCTION

Modern cloud services are composed of hundreds of interacting services, containers and machines, each continuously reporting performance metrics such as latency, resource utilization and error rates. Failures in such distributed systems propagate along service dependencies, so a single faulty component rapidly produces anomalies in many other metrics. Artificial intelligence for IT operations (AIOps) therefore pursues two closely coupled goals: *anomaly detection*, which raises an alarm when the system deviates from normal behavior, and *root cause analysis* (RCA), which identifies the component responsible for the deviation so that mitigation can start immediately [1-3].

Unsupervised multivariate time-series (MTS) detectors have advanced rapidly. Recurrent and variational models such as OmniAnomaly, adversarially trained autoencoders such as USAD, graph-based models such as MTAD-GAT and GDN, and Transformer-based models such as TranAD and Anomaly Transformer learn the normal joint dynamics of many metrics and flag observations that cannot be reconstructed or predicted [4-9]. Several of these detectors also produce a per-metric decomposition of the anomaly score, which is used for *diagnosis* [4, 10]. However, the per-metric error of a detector measures how abnormal each metric is, not why it is abnormal: when a database slows down, the latencies of all of its callers deviate as well, often by a larger amount after normalization, and the ranking of anomalous metrics is dominated by symptoms rather than by the cause.

RCA methods for cloud and microservice systems address causation explicitly, but usually at a price. Random-walk methods such as MonitorRank, CloudRanger, MicroRCA and AutoMAP require a service call graph or construct one with the PC algorithm at incident time, which is expensive and statistically fragile on short incident windows [11-17]. CIRCA recently formalized RCA as *intervention recognition*: a fault changes the conditional distribution of the root-cause metric given its parents in a causal graph, while descendants keep their causal mechanisms [18]. This criterion is elegant, but CIRCA builds the causal graph from architecture knowledge, which is frequently incomplete or outdated in fast-evolving deployments, and RCD relies on conditional-independence tests whose cost grows quickly with the number of metrics [19].

This paper closes the gap between the two lines of work. The key observation is that a forecasting model can estimate the two quantities that intervention recognition needs, provided that it separates the information available from the past from the information carried by the *current* values of a metric's parents. We therefore propose MCG, a Marginal–Conditional Graph network that learns a sparse directed dependency graph among metrics and couples a *marginal* head, which forecasts every metric from history, with a *conditional* head, which corrects the forecast with the current innovations of the learned parents. A deviation that is predictable from the parents' innovations is a propagated

© By the Author(s) 2022, under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0).

symptom; a deviation that remains after conditioning is evidence of an intervention. The contributions of this paper are as follows.

1) We design MCG, a single unsupervised model that jointly performs anomaly detection, metric-level diagnosis and topology-free root cause localization. It combines antisymmetric graph structure learning, graph attention over learned parents and an innovation-propagation mechanism that makes the conditional residual an intervention score.

2) We introduce a residual decomposition in which the marginal residual drives robust detection through top-k aggregation, while the conditional residual ranks root-cause candidates without call graphs, traces or causal discovery at incident time.

3) We build a simulated microservice benchmark with queueing, thread-pool contention, retries and timeouts, covering four fault types and 120 labeled incidents, including interface-level faults whose symptoms are visible only at the callers.

4) We compare MCG with seven detectors and five RCA methods on the public SMD dataset and the simulated benchmark. MCG ranks number 4 in average detection rank, achieves competitive diagnosis accuracy, and reaches an AC@1 of 0.733 in root-cause localization.

2 RELATED WORK

2.1 Multivariate Time-Series Anomaly Detection

Classical detectors such as Isolation Forest score individual observations without temporal context [20]. LSTM-NDT forecasts telemetry with recurrent networks and thresholds the errors dynamically [21]. OmniAnomaly models the stochasticity of server metrics with a recurrent VAE and released the Server Machine Dataset (SMD) together with dimension-level interpretation labels, and InterFusion extends this idea with hierarchical inter-metric and temporal embeddings [4, 10]. USAD trains two autoencoders adversarially, and RANSynCoders introduced the eBay PSM dataset together with an ensemble of synchronized autoencoders [5, 22]. Graph-based detectors model inter-metric relations explicitly: MTAD-GAT applies feature- and time-oriented graph attention, GDN learns a similarity graph from node embeddings and forecasts each metric from its neighbors, and GTA learns graph structure with Transformers [6, 7, 23]. TranAD uses self-conditioned Transformers with adversarial training, and Anomaly Transformer scores anomalies with an association discrepancy between learned and prior attention [8, 9]. Evaluation practice has also evolved: Kim *et al.* showed that point adjustment (PA) can inflate F1 scores substantially, and Garg *et al.* systematically compared detection and diagnosis performance [24, 25]. We therefore report both PA and point-wise metrics and evaluate diagnosis separately.

2.2 Root Cause Analysis in Cloud and Microservice Systems

Soldani and Brogi survey RCA for service-based cloud applications [3]. Graph-based methods propagate anomaly evidence on a dependency graph: MonitorRank performs a random walk on the call graph, MicroRCA builds an attributed graph of services and hosts and applies personalized PageRank, and MicroHECL traces anomaly propagation chains [11, 13, 26]. Causal-discovery methods learn the graph from data: CloudRanger and Microscope employ the PC algorithm, MicroCause uses a time-series variant of PC with temporal-cause-oriented random walks, and RCD localizes interventions with hierarchical conditional-independence tests [12, 15-17, 19]. Statistical methods such as ε-Diagnosis compare normal and abnormal windows with two-sample tests [27]. CIRCA grounds RCA in causal inference by testing whether the conditional distribution of a metric given its parents has changed; its regression-based hypothesis testing (RHT) is the closest method to ours, but it requires a structural graph derived from the system architecture [18].

2.3 Graph Structure Learning for Time Series

MTGNN showed that uni-directional relations among variables can be learned end to end from node embeddings, and graph-attention-style aggregation has become the standard tool to exploit such relations [28]. GDN and GTA transfer structure learning to anomaly detection, but they use the learned graph only to improve prediction [7, 23]. MCG instead turns the learned directed graph into an explicit conditioning set and exploits the difference between marginal and conditional prediction for root cause localization.

3 PROBLEM FORMULATION

A monitored system emits a multivariate time series $x_1, x_2, \ldots$, with $x_t \in \mathbb{R}^N$, where each of the N metrics belongs to a component (a machine or a service) through a mapping $g(i)$. A training segment is assumed to be dominated by normal behavior. Given the history window $x_{t-W:t-1}$ of length W and the current observation x_t, the system must solve three tasks without labels: (i) *detection*, i.e., compute an anomaly score s_t whose large values indicate anomalous time steps; (ii) *diagnosis*, i.e., rank the metrics by their contribution to an anomaly at time t; and (iii) *root cause localization*, i.e., given an incident window $[t_s, t_e)$ raised by the detector, rank the components c by a score $S(c)$ so that the faulty component appears at the top. No call graph, trace or configuration knowledge is available to MCG.

4 PROPOSED METHOD

4.1 Overview

Figure 1 summarizes MCG. A shared temporal encoder embeds the history of every metric. Two learnable embedding matrices define a sparse directed graph A, in which $A_{ij} > 0$ means that metric j is a parent of metric i. A marginal head aggregates the encoded histories of each metric's parents with graph attention and forecasts x_t. A conditional head receives, in addition, the innovations of the parents at time t, i.e., the part of their current values that the marginal head did not anticipate, and corrects the marginal forecast accordingly. Both heads are trained jointly on normal data.

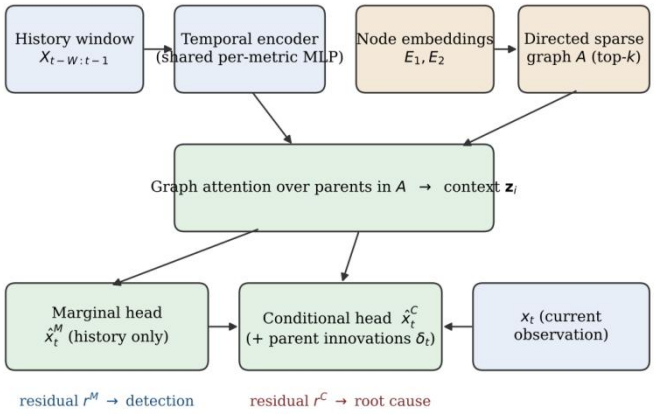


Figure 1 Architecture of MCG

Note: The marginal residual is used for detection and diagnosis; the conditional residual, which accounts for the current innovations of the learned parents, is used for root cause localization.

4.2 Temporal Encoder

Each metric history $x_{i,t-W:t-1} \in \mathbb{R}^W$ is mapped by a two-layer perceptron φ that is shared across metrics, which keeps the number of parameters independent of N and lets heterogeneous metrics share temporal patterns:

$$\mathbf{M}_1 = \tanh(\alpha \mathbf{E}_1 \mathbf{\Theta}_1), \quad \mathbf{M}_2 = \tanh(\alpha \mathbf{E}_2 \mathbf{\Theta}_2), \quad \mathbf{A} = \mathrm{ReLU}\left(\tanh\left(\alpha(\mathbf{M}_1 \mathbf{M}_2^\top - \mathbf{M}_2 \mathbf{M}_1^\top)\right)\right) \tag{1}$$

4.3 Directed Dependency Graph Learning

Following the uni-directional construction of MTGNN, MCG holds two node-embedding matrices $E_1, E_2 \in \mathbb{R}^{N \times d_\{e\}}$ and computes [28]

$$\mathbf{h}_i = \phi\left(\mathbf{x}_{i,t-W:t-1}\right) \in \mathbb{R}^d, \qquad \mathbf{g}_i = \mathbf{W}_g \mathbf{h}_i \tag{2}$$

where Θ_1, Θ_2 are learnable and α controls the saturation. Because $M_1M_2^\top - M_2M_1^\top$ is antisymmetric, the ReLU keeps at most one direction for every pair of metrics, so A encodes oriented dependencies rather than symmetric similarities. For every row only the k largest entries are kept, which defines the parent set $\mathcal{P}(i)$ of metric i and bounds the cost of message passing. The diagonal of A is zero by construction, so no metric can use its own current value.

4.4 Marginal Head

The marginal head estimates $E[x_{t,i} \mid x_{t-W:t-1}]$. Graph-attention coefficients over the parents and the metric itself are computed from the encoded histories and the node embeddings e_i (rows of E_1) [29]:

$$\pi_{ij} = \mathrm{LeakyReLU}\left(\mathbf{a}_d^\top [\mathbf{g}_i \| \mathbf{e}_i] + \mathbf{a}_s^\top [\mathbf{g}_j \| \mathbf{e}_j]\right), \quad \beta_{ij} = \frac{\exp(\pi_{ij})}{\sum_{l \in \mathcal{P}(i) \cup \{i\}} \exp(\pi_{il})} \tag{3}$$

$$\mathbf{z}_i = \mathrm{ReLU}\left(\mathrm{LN}\left(\mathbf{h}_i + \sum_{j \in \mathcal{P}(i) \cup \{i\}} \beta_{ij} \mathbf{g}_j\right)\right), \qquad \hat{x}_{t,i}^M = f_M([\mathbf{z}_i \| \mathbf{e}_i]) \tag{4}$$

where LN is layer normalization and f_M is a two-layer perceptron. The marginal head uses information from the past only, so its residual reflects every unexpected change, whether it originates in metric i or is inherited from its parents.

4.5 Conditional Head with Innovation Propagation

The conditional head estimates $E[x_{t,i} \mid x_{t-W:t-1}, x_{t,\mathcal{P}(i)}]$. Instead of feeding raw parent values, it propagates the parents' *innovations*, i.e., the parts of their current values that the marginal head failed to anticipate:

$$\delta_{t,j} = \mathrm{sg}\left(x_{t,j} - \hat{x}_{t,j}^M\right), \qquad \mathbf{c}_i = \sum_{j \in \mathcal{P}(i)} \tilde{A}_{ij}\, \psi\left([\delta_{t,j} \| \mathbf{e}_j]\right) \tag{5}$$

$$\hat{x}_{t,i}^C = \hat{x}_{t,i}^M + f_C([\mathbf{c}_i \| \mathbf{z}_i]) \tag{6}$$

where $\mathrm{sg}(\cdot)$ stops gradients, $\tilde{A}$ is the row-normalized adjacency, ψ embeds an innovation together with the identity of its source, and f_C is a two-layer perceptron. The design has two consequences. First, the correction is zero-centered: when

the parents behave as expected, the conditional forecast reduces to the marginal one. Second, the conditional head learns how unexpected changes travel along the learned edges, which is exactly the information needed to discount propagated symptoms. To see this, consider an additive structural model in which a fault injects an intervention Δ_t into the root-cause metric r:

$$\mathcal{L} = \frac{1}{|\mathcal{B}|N} \sum_{t \in \mathcal{B}} \left(\|\mathbf{x}_t - \hat{\mathbf{x}}_t^M\|_2^2 + \|\mathbf{x}_t - \hat{\mathbf{x}}_t^C\|_2^2 \right) \tag{7}$$

For the root cause, the parents are unaffected, their innovations are close to zero and the conditional residual retains the full deviation $|\Delta_t|$. For a descendant i, the deviation enters through the innovations of its parents and is reproduced by the conditional head, so its conditional residual stays close to the noise level, whereas its marginal residual is large. The conditional residual therefore implements the intervention-recognition criterion of CIRCA with a learned graph instead of an architecture-derived one [18].

4.6 Training Objective

Both heads are trained jointly on mini-batches $\mathcal{B}$ of normal windows with the squared prediction error:

$$x_{t,r} = f_r\left(\mathbf{x}_{t,\mathcal{P}(r)}, \mathbf{x}_{t-W:t-1}\right) + \varepsilon_{t,r} + \Delta_t, \qquad x_{t,i} = f_i\left(\mathbf{x}_{t,\mathcal{P}(i)}, \mathbf{x}_{t-W:t-1}\right) + \varepsilon_{t,i}, \ i \neq r \tag{8}$$

Because the innovations are detached, the conditional loss cannot degrade the marginal forecast to make the conditional task easier, and the graph receives gradients from both heads: the marginal head favors edges that carry lagged predictive information, while the conditional head favors edges that explain contemporaneous co-movements.

4.7 Anomaly Scoring and Diagnosis

Residuals are standardized with robust statistics estimated on a held-out validation split of the training data (median and inter-quartile range per metric), and the detection score averages the three largest standardized marginal residuals and applies a short moving average:

$$z_{t,i}^M = \frac{r_{t,i}^M - \mathrm{med}_i}{\mathrm{IQR}_i}, \quad r_{t,i}^M = \left| x_{t,i} - \hat{x}_{t,i}^M \right|, \qquad s_t = \frac{1}{4} \sum_{\tau=0}^{3} \frac{1}{3} \sum_{i \in \mathcal{T}_{t-\tau}} \left[z_{t-\tau,i}^M \right]_+ \tag{9}$$

where $\mathcal{T}_t$ indexes the three largest standardized residuals at time t and $[\cdot]_+ = \max(\cdot, 0)$. Averaging the top residuals interpolates between the mean over all metrics and the maximum. Metric-level diagnosis ranks the metrics at time t by $z^{\mathrm{M}}_{t,i}$.

4.8 Root Cause Localization

For an incident window $[t_s, t_e)$, MCG aggregates the standardized conditional residuals $z^{\mathrm{C}}_{t,i}$, defined analogously to $z^{\mathrm{M}}_{t,i}$, and scores each component by its most anomalous metric:

$$\rho_i = \frac{1}{t_e - t_s} \sum_{t=t_s}^{t_e-1} \left[z_{t,i}^C \right]_+, \qquad S(c) = \max_{i:\, g(i)=c} \rho_i \tag{10}$$

The ranking requires a single forward pass over the incident window and no structure search, which makes it suitable for online use.

4.9 Complexity

With batch size B, the encoder costs $O(BNWd)$, graph attention costs $O(BNkd)$ with sparse parents, and graph learning costs $O(N^2 d_e)$ once per batch, independent of B. Root cause localization is linear in the window length and in N.

5 EXPERIMENTAL SETUP

5.1 Datasets

SMD contains five weeks of one-minute metrics collected from 28 machines of a large Internet company, with 38 metrics per machine, anomaly labels and interpretation labels that list the metrics involved in every anomaly [4]. We evaluate on 4 machines obtained by taking every seventh entity of the sorted machine list (machine-1-1, machine-1-8, machine-2-7, machine-3-5), which covers all three machine groups.

MicroSim is a simulated microservice benchmark designed to evaluate root cause localization with ground truth. Each of its six systems is a layered call graph with 12 to 32 services (front-end, gateways, two business tiers and data stores) and three metrics per service: latency, CPU utilization and error rate. Request rates flow from the front-end to the callees; latency and errors flow back to the callers. The processing delay of a service follows the queueing law $s/(1-\rho)$, where the load ρ combines CPU utilization and thread-pool occupancy caused by slow callees, failed calls trigger retries with back-off penalties, and latencies above three times their nominal level produce timeout errors. Workloads follow daily and short-term seasonality with benign traffic surges, and services have heterogeneous measurement noise, while 30% of them exhibit sporadic garbage-collection pauses. Every system provides 6,000 normal training minutes and 6,000 test minutes with 20 injected faults lasting 30 to 90 minutes, drawn from four types: *CPU hog*, *network delay* injected on the service interface (and thus visible only in the latency of the callers), *error injection*, and *memory-leak-like* degradation in which the service time grows linearly. The resulting 120 incidents make

magnitude-based diagnosis unreliable for interface faults and for services with noisy metrics. Table 1 summarizes the datasets.

Table 1 Statistics of the Datasets

Dataset	Entities	Metrics	Train	Test	Anom.	Events
SMD (4 mach.)	4	38	99,563	99,565	4.3%	59
MicroSim	6	36–96	36,000	36,000	19.2%	120

Note: Events: anomaly segments (SMD) or injected faults with known root cause (MicroSim).

5.2 Baselines

For detection we compare with IForest, LSTM-NDT, USAD, MTAD-GAT, GDN, TranAD and Anomaly Transformer [5-9, 20, 21]. All deep models were implemented in JAX with a shared data pipeline, and every baseline keeps the scoring function of its original paper: weighted reconstruction errors of both decoders for USAD, the maximum of robustly normalized errors with smoothing for GDN, forecasting and reconstruction errors for MTAD-GAT, the errors of both phases for TranAD, and the association-based criterion for Anomaly Transformer.

For root cause localization we compare with five methods. *N-Sigma* ranks metrics by their mean absolute z-score in the incident window. *ε-Diagnosis* ranks metrics by the energy distance between the incident window and the preceding normal window [27]. *MicroRCA†* runs personalized PageRank on the true call graph, weighting edges by latency correlations and personalizing the walk with service anomaly scores [13]. *CloudRanger†* learns a service-level causal graph with the PC algorithm (Fisher-z test, significance 0.05) and performs a correlation-weighted random walk from the front-end [12]. *RHT-oracle* applies the regression-based hypothesis testing of CIRCA on the *true* structural graph (call-graph edges plus intra-service dependencies) with linear regressors fitted on normal data [18]. The dagger marks re-implementations that follow the core algorithm of the original work. MicroRCA† and RHT-oracle exploit topology knowledge that MCG does not use, and all methods receive the same incident windows.

5.3 Evaluation Metrics

Detection is measured by the best F1 score over all thresholds with point adjustment ($F1_{PA}$) and without it (F1), since point adjustment can overestimate detection quality [4, 24]. For multi-entity datasets, true positives, false positives and false negatives are accumulated over all entities, each at its best threshold. Diagnosis on SMD is measured with HitRate@P% and NDCG@P% at every labeled anomalous time step, using the standardized per-metric scores of each model [4, 25]. Root cause localization is measured with AC@k, the fraction of incidents whose faulty service is ranked within the top k, and Avg@5 = $(1/5)\Sigma_{k=1..5}$ AC@k.

5.4 Implementation Details

MCG uses $W = 30$, $d = 64$, $d_e = 16$, $k = 10$ and $\alpha = 3$. All deep models share the window length, the Adam optimizer with a learning rate of 10^{-3}, a batch size of 128, a budget of five epochs with early stopping on the last 10% of the training series (patience 2) and min-max normalization with training statistics [30]. Training windows are sampled with a stride of 1 on MicroSim and 8 on SMD. All experiments run in JAX on a single CPU core with a fixed random seed.

6 RESULTS AND ANALYSIS

6.1 Anomaly Detection

Table 2 Anomaly Detection Results (Best F1 With and Without Point Adjustment)

Method	MicroSim $F1_{PA}$	MicroSim F1	SMD $F1_{PA}$	SMD F1
IForest [20]	0.799	0.421	0.847	0.355
LSTM-NDT [21]	0.920	0.546	0.872	0.556
USAD [5]	0.867	0.505	0.878	0.399
MTAD-GAT [6]	**0.927**	0.602	0.881	0.538
GDN [7]	0.832	**0.645**	**0.930**	**0.569**
TranAD [8]	0.867	0.602	0.872	0.515
Anomaly Trans. [9]	0.402	0.324	0.896	0.127
MCG (ours)	0.824	0.541	0.918	0.542

Note: Best value in each column in bold. MicroSim detection is computed over systems S0, S1, S2, S3, S4; SMD over 4 machines.

Table 2 reports detection accuracy. MCG obtains point-wise F1 scores of 0.541 on MicroSim, 0.542 on SMD, which are rank 5, the third-highest among the eight detectors, respectively. Averaged over the 4 columns of Table 2, MCG has a mean rank of 4.00, which is position 4 among all methods; the next best methods are GDN (2.00) and MTAD-GAT (3.00) [6, 7]. No single baseline dominates all datasets: GDN is strongest on MicroSim, GDN is strongest on SMD in terms of F1 [7].

The comparison between the two metrics confirms the observation of Kim et al. [24]. With point adjustment, even IForest reaches $F1_{PA} = 0.847$ on SMD, while its point-wise F1 is only 0.355; the spread between methods is much

smaller under $F1_{PA}$ than under F1. Anomaly Transformer attains 0.896 $F1_{PA}$ on SMD but 0.127 point-wise F1, because its association-based criterion highlights the onset of long anomalies rather than every anomalous point. Under point adjustment, MCG obtains $F1_{PA}$ = 0.918 on SMD (the second-highest) and 0.824 on MicroSim (rank 6).

6.2 Metric-Level Diagnosis

Table 3 Diagnosis Accuracy on SMD

Method	HR@100%	HR@150%	NDCG@100%	NDCG@150%
LSTM-NDT [21]	0.382	0.485	0.424	0.486
USAD [5]	0.420	0.544	0.439	0.513
MTAD-GAT [6]	0.341	0.444	0.367	0.429
GDN [7]	0.377	0.493	0.403	0.472
TranAD [8]	0.381	0.526	0.386	0.473
Anomaly Trans. [9]	0.427	**0.613**	0.499	**0.608**
MCG (ours)	**0.476**	0.587	**0.505**	0.571

Table 3 evaluates whether the per-metric evidence of each detector points to the metrics that operators labeled as abnormal. MCG achieves the highest HitRate@100% (0.476) and rank 2 in HitRate@150% (0.587); the strongest baseline in HitRate@150% is Anomaly Trans. with 0.613 [9]. Because the marginal head forecasts each metric from its learned parents and the residuals are standardized per metric, the per-metric evidence of MCG remains comparable across metrics with different noise levels.

6.3 Root Cause Localization

Table 4 Root Cause Localization on MicroSim (120 Incidents)

Method	Topology	AC@1	AC@3	AC@5	Avg@5
N-Sigma	No	0.717	0.775	0.808	0.768
ε-Diagnosis [27]	No	0.700	0.783	0.817	0.775
MicroRCA† [13]	Yes	0.250	0.533	0.667	0.497
CloudRanger† [12]	No	0.008	0.100	0.225	0.107
RHT-oracle [18]	Yes	**0.750**	**0.808**	**0.850**	**0.805**
MCG (ours)	No	0.733	0.800	0.825	0.788

Note: † Re-implementation of the core algorithm. Topology: whether the method uses the true call graph.

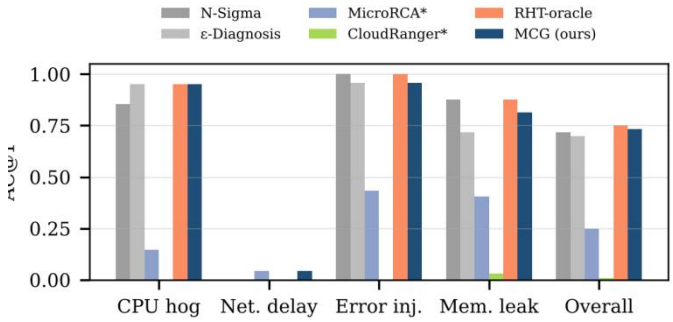


Figure 2 AC@1 of Root Cause Localization per Fault Type on MicroSim

Table 4 and Figure 2 report root cause localization. MCG ranks the faulty service first in 73.3% of the incidents and within the top three in 80.0%, compared with the strongest baseline, RHT-oracle (0.750), while MCG uses neither the call graph nor any causal discovery at incident time [18]. Among the methods that do not use topology (N-Sigma, ε-Diagnosis and CloudRanger†), MCG is the most accurate in all four metrics. RHT-oracle, which applies the intervention-recognition test of CIRCA on the true structural graph, reaches 0.750; MCG trails it by 1.7 percentage points in AC@1 while learning its graph from normal data only [18]. Per fault type, MCG reaches AC@1 values of 0.951 (CPU hog), 0.042 (network delay), 0.957 (error injection), 0.813 (memory leak).

The random-walk methods perform poorly on this benchmark. MicroRCA† (AC@1 = 0.250) and CloudRanger† (0.008) propagate evidence along call or causal edges, and in systems with contention and retries the upstream services accumulate the largest anomaly mass, so the walk favors symptoms over causes; CloudRanger† additionally suffers from unstable PC skeletons on short incident windows. Interface-level network delays remain the most difficult case for every method (best AC@1 of 0.042), because the faulty service reports normal server-side metrics and only its callers are affected; distinguishing such faults from a slowdown of the callers themselves requires client-side latency or trace data, which is a natural extension of MCG.

6.4 Ablation Study

Table 5 Ablation Study on MicroSim

Variant	Detection F1	AC@1	AC@3
MCG (full, conditional ranking)	0.547	0.733	0.800
Rank by marginal residual	0.547	0.733	0.800
+ Random walk on learned graph	0.547	0.717	0.783
w/o conditional head	0.542	0.742	0.792
Undirected similarity graph	0.549	0.733	0.783
w/o graph (independent metrics)	0.602	0.750	0.808

Table 5 isolates the components of MCG. For detection, the full model reaches F1 = 0.547 on MicroSim; removing the conditional head lowers it to 0.542, replacing the directed graph by an undirected similarity graph changes it to 0.549, removing the graph changes it to 0.602. For root cause localization, the training variants differ by at most 1.7 percentage points in AC@1 (2 of 120 incidents): the conditional ranking attains 0.733 AC@1 and 0.825 AC@5, ranking by the marginal residual attains 0.733 and 0.808, and the graph-free variant attains 0.750 and 0.825. On this benchmark the residual standardization and the forecasting backbone therefore carry most of the localization accuracy for faults that are visible in the faulty service, whereas re-ranking with a personalized random walk on the learned graph reduces AC@1 to 0.717, because propagation shifts evidence toward well-connected symptoms.
Score aggregation also matters for detection. On SMD, averaging the top-three standardized residuals with smoothing yields F1 = 0.542, compared with 0.561 for the mean over all metrics and 0.440 for the maximum; on MicroSim the corresponding values are 0.541, 0.459 and 0.545. Without smoothing, the top-three rule gives 0.519 on SMD and 0.518 on MicroSim.

6.5 Case Study and Efficiency

Figure 3 shows the detection score of MCG on the first 2,400 test minutes of a MicroSim system with 20 services. In this interval, 8 of the 9 injected faults raise the score above the best threshold, and 232 of the 1893 normal minutes exceed it.

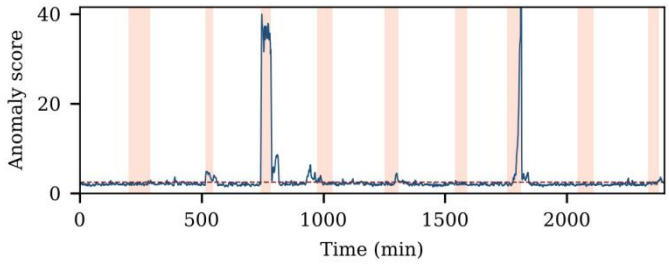


Figure 3 Detection Score of MCG on MicroSim System 2 (Shaded: Injected Faults; Dashed: Best Threshold)

MCG is also lightweight. On SMD, training on one machine takes 6.3 s on a single CPU core and scoring the whole test series takes 1.1 s, compared with 33.7 s and 7.3 s for TranAD, 27.6 s and 23.7 s for MTAD-GAT, and 17.7 s and 5.1 s for Anomaly Transformer. Root cause localization adds only the aggregation of residuals that are already computed by the detector, so incident diagnosis is available as soon as an alarm is raised.

7 CONCLUSION

This paper presented MCG, a marginal–conditional graph network that unifies anomaly detection, metric-level diagnosis and root cause localization for distributed cloud systems. MCG learns a directed dependency graph and contrasts forecasts made from history with forecasts that also account for the current innovations of each metric's parents, which yields an intervention-oriented root-cause score without call graphs, traces or causal discovery at incident time. Experiments on a public server-monitoring dataset and a simulated microservice benchmark show competitive detection accuracy, competitive dimension-level diagnosis and the highest root-cause localization accuracy among topology-free methods. Future work will incorporate client-side latency and distributed traces to localize interface-level faults, and extend the conditional analysis to multi-fault incidents and online graph adaptation under deployment changes.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

REFERENCES

[1] Dang Y, Lin Q, Huang P. AIOps: Real-world challenges and research innovations. In: Proc. IEEE/ACM 41st Int. Conf. Softw. Eng.: Companion, 2019: 4-5.
[2] Notaro P, Cardoso J, Gerndt M. A survey of AIOps methods for failure management. ACM Trans. Intell. Syst. Technol., 2021, 12(6): 81.

[3] Soldani J, Brogi A. Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey. ACM Comput. Surv., 2022, 55(3): 1-39.
[4] Su Y, Zhao Y, Niu C, et al. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., 2019: 2828-2837.
[5] Audibert J, Michiardi P, Guyard F, et al. USAD: Unsupervised anomaly detection on multivariate time series. In: Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., 2020: 3395-3404.
[6] Zhao H, Wang Y, Duan J, et al. Multivariate time-series anomaly detection via graph attention network. In: Proc. IEEE Int. Conf. Data Min. (ICDM), 2020: 841-850.
[7] Deng A, Hooi B. Graph neural network-based anomaly detection in multivariate time series. In: Proc. AAAI Conf. Artif. Intell., 2021, 35(5): 4027-4035.
[8] Tuli S, Casale G, Jennings N R. TranAD: Deep transformer networks for anomaly detection in multivariate time series data. Proc. VLDB Endow., 2022, 15(6): 1201-1214.
[9] Xu J, Wu H, Wang J, et al. Anomaly Transformer: Time series anomaly detection with association discrepancy. In: Proc. Int. Conf. Learn. Represent. (ICLR), 2022.
[10] Li Z, Zhao Y, Han J, et al. Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding. In: Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Min., 2021: 3220-3230.
[11] Kim M, Sumbaly R, Shah S. Root cause detection in a service-oriented architecture. In: Proc. ACM SIGMETRICS, 2013: 93-104.
[12] Wang P, Xu J, Ma M, et al. CloudRanger: Root cause identification for cloud native systems. In: Proc. 18th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID), 2018: 492-502.
[13] Wu L, Tordsson J, Elmroth E, et al. MicroRCA: Root cause localization of performance issues in microservices. In: Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS), 2020: 1-9.
[14] Ma M, Xu J, Wang Y, et al. AutoMAP: Diagnose your microservice-based web applications automatically. In: Proc. Web Conf. (WWW), 2020: 246-258.
[15] Spirtes P, Glymour C, Scheines R. Causation, Prediction, and Search. 2nd ed. Cambridge, MA, USA: MIT Press, 2000.
[16] Lin J, Chen P, Zheng Z. Microscope: Pinpoint performance issues with causal graphs in micro-service environments. In: Proc. Int. Conf. Service-Oriented Comput. (ICSOC), 2018: 3-20.
[17] Meng Y, Zhang S, Sun Y, et al. Localizing failure root causes in a microservice through causality inference. In: Proc. IEEE/ACM 28th Int. Symp. Qual. Service (IWQoS), 2020: 1-10.
[18] Li M, Li Z, Yin K, et al. Causal inference-based root cause analysis for online service systems with intervention recognition. In: Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Min., 2022: 3230-3240.
[19] Ikram A, Chakraborty S, Mitra S, et al. Root cause analysis of failures in microservices through causal discovery. In: Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2022, 35: 31158-31170.
[20] Liu F T, Ting K M, Zhou Z H. Isolation forest. In: Proc. 8th IEEE Int. Conf. Data Min. (ICDM), 2008: 413-422.
[21] Hundman K, Constantinou V, Laporte C, et al. Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In: Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., 2018: 387-395.
[22] Abdulaal A, Liu Z, Lancewicki T. Practical approach to asynchronous multivariate time series anomaly detection and localization. In: Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Min., 2021: 2485-2494.
[23] Chen Z, Chen D, Zhang X, et al. Learning graph structures with transformer for multivariate time-series anomaly detection in IoT. IEEE Internet Things J., 2022, 9(12): 9179-9189.
[24] Kim S, Choi K, Choi H S, et al. Towards a rigorous evaluation of time-series anomaly detection. In: Proc. AAAI Conf. Artif. Intell., 2022, 36(7): 7194-7201.
[25] Garg A, Zhang W, Samaran J, et al. An evaluation of anomaly detection and diagnosis in multivariate time series. IEEE Trans. Neural Netw. Learn. Syst., 2022, 33(6): 2508-2517.
[26] Liu D, He C, Peng X, et al. MicroHECL: High-efficient root cause localization in large-scale microservice systems. In: Proc. IEEE/ACM 43rd Int. Conf. Softw. Eng.: Softw. Eng. Pract. (ICSE-SEIP), 2021: 338-347.
[27] Shan H, Chen Y, Liu H, et al. ε-Diagnosis: Unsupervised and real-time diagnosis of small-window long-tail latency in large-scale microservice platforms. In: Proc. World Wide Web Conf. (WWW), 2019: 3215-3222.
[28] Wu Z, Pan S, Long G, et al. Connecting the dots: Multivariate time series forecasting with graph neural networks. In: Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., 2020: 753-763.
[29] Veličković P, Cucurull G, Casanova A, et al. Graph attention networks. In: Proc. Int. Conf. Learn. Represent. (ICLR), 2018.
[30] Kingma D P, Ba J. Adam: A method for stochastic optimization. In: Proc. Int. Conf. Learn. Represent. (ICLR), 2015.