Hyperbolic Graph Attention with Confidence-Gated Agent Reranking for Cloud-Edge Microservice Root Cause Localization

Yechen He*, Chenyuan Feng†, Nuocheng Yang‡, Zihan Chen§, Tony Q. S. Quek§,
* State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China
† College of Computer Science, University of Exeter, U.K.
‡ Beijing Laboratory of Advanced Information Network, Beijing University of Posts and Telecommunications, China
§ Information Systems Technology and Design Pillar, Singapore University of Technology and Design, Singapore
Email: c.feng@exeter.ac.uk

***Abstract*—Root cause localization is essential for the service reliability and QoS guarantee of cloud-edge microservice systems, where multiple systems are increasingly co-deployed on shared infrastructure to improve resource utilization. Such hybrid deployments expose two structural weaknesses of existing methods. First, service dependencies form a tree-like hierarchy across cloud and edge tiers whose exponential branching cannot be faithfully embedded in Euclidean space, leading to distorted neighborhoods and indecisive rankings. Second, gateway services that aggregate traffic from co-located systems accumulate amplified anomaly signals and are systematically misidentified as culprits. Motivated by these two failure modes, we propose hyperbolic-agent graph reasoning, which jointly addresses geometric distortion and gateway-induced false alarms. A hyperbolic constrained spatio-temporal graph attention network projects node features onto the Poincaré ball and aggregates evidence at the network-segment level to produce a hierarchy-aware ranking. A confidence-gated agent then monitors the score margin between top candidates and selectively invokes large language model to refine the ranking when the margin is narrow, while a symbolic gateway-suppression rule corrects residual false alarms on traffic-aggregating services. Experiments on two cloud-edge benchmarks show that HAGR reaches ACC@1 of 59.3% and 70.8%, outperforming the second-best method by 16.3% and 7.2%.**

***Index Terms*—Microservice, QoS Modeling, Root Cause Localization, Hyperbolic Graph Learning, AI Agent.**

I. Introduction

Maintaining Quality of Service (QoS) and reliability in cloud-edge microservice systems is critical for modern distributed applications. Modern software systems increasingly adopt microservice architectures that span a cloud-edge continuum [1]. This shift brings computation closer to users and improves responsiveness, but it also reshapes the failure surface. When a fault occurs, abnormal signals propagate through service dependencies and shared resources, and symptoms may appear on both application calls and kernel-level resources [2]. Operators are expected to locate the root cause quickly, but they often have limited labels and incomplete causal traces. Fig. 1 illustrates the setting we target, where a root-cause failure propagates across service dependencies under unstable cloud-edge connectivity.

Compared with single-cluster settings, cloud-edge operations exhibit two intrinsic properties that exacerbate root cause localization. First, operators often co-deploy multiple microservice systems on shared infrastructure for higher utilization, a setting we refer to as hybrid deployment. In this regime, metrics from non-causal systems inject noise into every failure window, complicating both anomaly detection and ranking. Second, service topologies form a tree-like hierarchy in which a few edge ingress points fan out through business logic into a dense cloud layer of databases and caches. Gateway and ingress services sit at the entry of multiple systems and thus accumulate anomalies even when they are not the fault origin, an asymmetry that frequently misleads diagnostic algorithms into flagging them as culprits.

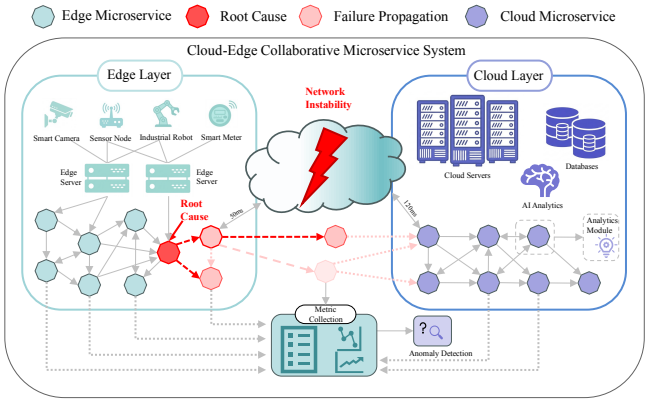

Fig. 1: An example of cloud-edge collaborative microservice system architecture.

Existing root cause localization methods tackle these challenges in isolation rather than jointly. Random-walk approaches such as CloudRanger [3] and MicroRCA [4] build correlation or PageRank-style graphs under stable-neighborhood assumptions and degrade once hybrid deployment injects metric noise. Causal inference methods such as CausIL [5] and CausalRCA [6] rely on fixed statistical thresholds that are sensitive to cross-segment latency fluctuations. Graph neural approaches, notably MicroCERCL [7], model heterogeneous dependencies with failure backpropagation and achieve the strongest reported accuracy, yet embed topology in Euclidean space whose polynomial volume growth cannot host exponentially branching hierarchies without distortion [8]. A parallel

Corresponding author: Chenyuan Feng.

line of work applies language-model agents to root cause analysis [9]–[11], bringing symbolic reasoning at the cost of high inference latency and susceptibility to hallucination when upstream evidence is unreliable.

These observations reveal two fundamental gaps that account for most of the residual error in this setting, which we formalize as the following challenges.

C1: Microservice dependencies grow exponentially with depth, yet Euclidean graph attention flattens such hierarchies, blurring culprits and benign downstream neighbors and bounding what Euclidean embedding can achieve.

C2: Under weak evidence, graph neural networks can only assign marginally different scores to multiple candidates, allowing gateway services to seize the top position through traffic-aggregated symptoms that neural scoring alone cannot distinguish from genuine causal evidence.

To close both gaps, we propose hyperbolic-agent graph reasoning (HAGR). HAGR first encodes heterogeneous topology snapshots through a hyperbolic constrained spatio-temporal graph attention network (HCSTGAN), which projects node features onto the Poincaré ball to preserve hierarchical relations and pools evidence at the network-segment level to reflect the regional concentration of cloud-edge deployments. A structured-prompt agent reranker is then activated only when the top-two score margin is narrow or the leading candidate matches a known victim pattern, after which a symbolic gateway-suppression rule eliminates residual false alarms on traffic-aggregating services.

The main contributions of this paper are as follows.

- We design HCSTGAN, a graph attention network that addresses hierarchical distortion in cloud-edge microservice topologies by embedding heterogeneous dependencies on the Poincaré ball and pairing it with a network-segment attention pooling module. This captures the regional concentration of cloud-edge deployments and reduces the structural distortion of Euclidean models **(for C1)**.
- We propose a confidence-gated agent reranking mechanism that activates a language-model reasoner only when the graph neural network is indecisive. A structured prompt guides the agent through resource-based reasoning, and a symbolic gateway-suppression rule then corrects residual false alarms on traffic-aggregating services **(for C2)**.
- We evaluate HAGR on two hybrid-deployed cloud-edge benchmarks. HAGR reaches ACC@1 of 59.3% and 70.8%, which improves over the strongest prior method MicroCERCL by 16.3% and 7.2%, with consistent gains on ACC@K and MRR.

II. Problem Formulation

A. System Model and Problem Statement

We represent a microservice system at time interval t as a heterogeneous graph $G_t = (\mathcal{V}, \mathcal{E}, \mathcal{D})$. The node set $\mathcal{V} = \mathcal{V}_s \cup \mathcal{V}_p \cup \mathcal{V}_n$ consists of services $\mathcal{V}_s$, instances $\mathcal{V}_p$, and physical nodes $\mathcal{V}_n$. The edge set $\mathcal{E}$ contains direct invocation dependencies between services and indirect resource-sharing dependencies between instances and physical nodes. The set $\mathcal{D}$ indexes network segments that distinguish cloud and edge domains. Auto-scaling and network fluctuations cause topology to change over time, so we organize snapshots into a dynamic topology stack $\mathcal{G} = \{G_{t_1^d}, G_{t_2^d}, \ldots, G_{t_l^d}\}$, where each $G_{t_i^d}$ is stable during the interval $[t_i^d, t_{i+1}^d)$. Each node $v \in \mathcal{V}$ carries a time-series feature vector $f_v^{t_n}$ composed of latency, CPU, memory, and network metrics collected at timestamp t_n.

Given the topology stack $\mathcal{G}$, the node features $\{f_v^{t_n}\}$, and a set of anomalous nodes $\mathcal{A} \subseteq \mathcal{V}$ supplied by an upstream anomaly detector, the goal of root cause localization is to learn a ranking function

$$R : \mathcal{V} \to [0, 1] \tag{1}$$

that assigns each node a probability of being the root cause. The final output is a list sorted by R. A key property we exploit is failure backpropagation: when service A invokes service B and B fails, A also exhibits anomalous behavior. This property, combined with topology aggregation within network segments, forms the basis of our design.

B. Hyperbolic Geometry Preliminaries

Hyperbolic space is a natural geometry for hierarchical structures because its volume grows exponentially with radius, matching the branching pattern of service call graphs [8].In contrast, the polynomial volume growth of Euclidean space cannot accommodate exponentially branching hierarchies without distorting pairwise distances and collapsing parent child relations. We adopt the Poincaré ball model $\mathbb{B}_c^d = \{x \in \mathbb{R}^d : c\|x\|^2 < 1\}$ with curvature $-1/c$. Two operations are used in Sec. III to move between Euclidean space and this manifold. The exponential map at the origin projects a tangent vector $v \in \mathbb{R}^d$ onto the ball:

$$\exp_0^c(v) = \tanh(\sqrt{c}\|v\|) \cdot \frac{v}{\sqrt{c}\|v\|}. \tag{2}$$

The logarithmic map at the origin projects a point $y \in \mathbb{B}_c^d$ back to the tangent space:

$$\log_0^c(y) = \operatorname{arctanh}(\sqrt{c}\|y\|) \cdot \frac{y}{\sqrt{c}\|y\|}. \tag{3}$$

HCSTGAN uses $\exp_0^c$ to embed node features onto the Poincaré ball for geometric regularization, and $\log_0^c$ to recover tangent-space representations for downstream attention pooling.

III. Methodology

A. Overview

Fig. 2 presents the architecture of HAGR. The framework has three stages. Anomaly detection scans time-series metrics and produces a set of candidate nodes $\mathcal{A} \subseteq \mathcal{V}$. We adopt the Birch clustering detector [12] for this stage. HCSTGAN then takes the dynamic topology stack $\mathcal{G}$ and $\mathcal{A}$ as input and produces an initial root cause ranking through hyperbolic graph attention. Finally, the confidence-gated agent reranking module

examines the score margin at the top of the ranking and invokes a language-model reasoner only when the margin is small.

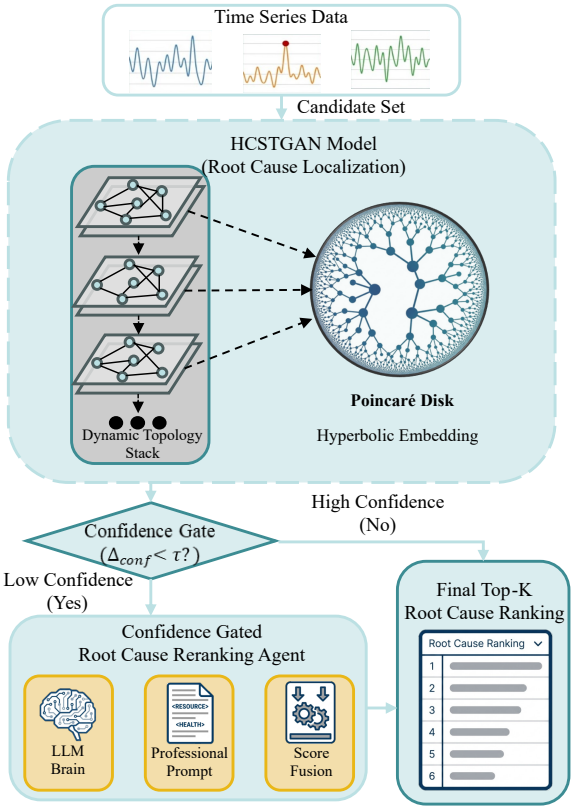

Fig. 2: Overview of HAGR framework for root cause localization in cloud-edge microservices.

B. Hyperbolic Constrained Spatio-Temporal Graph Attention Network

HCSTGAN integrates spatial attention with temporal recurrence. Its two key designs are a hyperbolic transformation that regularizes features on the Poincaré ball, and a network-segment attention pooling (NSAP) module that reflects the regional structure of cloud-edge deployments.

1) Heterogeneous Graph Attention: For each topology snapshot $G_{t_l^d}$ in $\mathcal{G}$, we perform heterogeneous graph attention over different edge types. The initial feature $h_v^{(0)}$ is the time-series metric vector $f_v^{t_n}$. For a node v and a type-r neighbor u, each attention head $k \in \{1, \ldots, K\}$ uses a learnable weight $W_r^{(k)} \in \mathbb{R}^{d' \times d}$ to project features into $z_v^{(k)} = W_r^{(k)} h_v^{(l-1)}$ and $z_u^{(k)} = W_r^{(k)} h_u^{(l-1)}$. Attention coefficients over the type-r neighborhood $\mathcal{N}_r(v)$ are

$$\alpha_{vu}^{(k)} = \frac{\exp(\text{LeakyReLU}((a^{(k)})^\top [z_v^{(k)} \| z_u^{(k)}]))}{\sum_{u' \in \mathcal{N}_r(v)} \exp(\text{LeakyReLU}((a^{(k)})^\top [z_v^{(k)} \| z_{u'}^{(k)}]))}, \tag{4}$$

where $a^{(k)} \in \mathbb{R}^{2d'}$ is a learnable vector and $\|$ denotes concatenation. The updated feature concatenates outputs from all heads:

$$h_v^{(l)} = \Big\|_{k=1}^{K} \sigma\left(\sum_{u \in \mathcal{N}_r(v)} \alpha_{vu}^{(k)} z_u^{(k)} \right). \tag{5}$$

Features from different node types are stacked into a single matrix $H \in \mathbb{R}^{|\mathcal{V}| \times d_{\text{out}}}$.

2) Hyperbolic Space Transformation: Cloud-edge service topologies branch exponentially with depth, and Euclidean embeddings distort such hierarchies because their volume grows only polynomially with radius. To mitigate this distortion, we project features onto the Poincaré ball as a geometric regularizer. With curvature $c = 1$, each row h_v of H is first mapped onto the ball through the exponential map in Eq. (2):

$$h_v^{\mathbb{B}} = \tanh(\sqrt{c}\|h_v\|) \cdot \frac{h_v}{\sqrt{c}\|h_v\| + \epsilon}, \tag{6}$$

where ϵ is a small constant for numerical stability. The logarithmic map then brings the features back to the tangent space:

$$h_v^{T} = \operatorname{arctanh}(\sqrt{c}\|h_v^{\mathbb{B}}\|) \cdot \frac{h_v^{\mathbb{B}}}{\sqrt{c}\|h_v^{\mathbb{B}}\| + \epsilon}. \tag{7}$$

Although $\log_0^c \circ \exp_0^c$ is mathematically an identity in the forward pass, the detour through the bounded Poincaré ball reshapes the loss landscape and yields gradients that pull hierarchically distant nodes apart during training. The bounded geometry of the Poincaré ball matches the tree-like branching of service call graphs and helps the model preserve hierarchical structure. The transformed matrix is $\tilde{H} = \text{ReLU}([h_1^T; \ldots; h_{|\mathcal{V}|}^T]^\top)$.

3) Network-Segment Attention Pooling: Services in cloud-edge deployments cluster within network segments, and failure propagation produces characteristic aggregation patterns at the segment level. NSAP captures this structure through a two-level attention. We first compute a node-level attention gate:

$$g_v = \text{softmax}_{\mathcal{V}}(W_g \tilde{H}_v), \tag{8}$$

where W_g is a learnable weight matrix. The gate g_v drives two parallel computations. It produces a graph-level readout that summarizes the overall snapshot state:

$$r = \sum_{v \in \mathcal{V}} g_v \odot \tilde{H}_v. \tag{9}$$

It also feeds into segment-level aggregation. For each network segment $d \in \mathcal{D}$ with node set $\mathcal{V}_d$, the segment embedding is

$$c_d = \frac{1}{|\mathcal{V}_d|} \sum_{v \in \mathcal{V}_d} g_v. \tag{10}$$

Stacking these into $C = [c_1; \ldots; c_{|\mathcal{D}|}]^\top$, we apply scaled dot-product cross-attention $CW_Q(CW_K)^\top$ followed by softmax to obtain refined segment embeddings $\hat{c}_d$. The segment context is then propagated back to each node:

$$a_v = \max_{t_l^d} \left(g_v^{(t_l^d)} \odot \text{ReLU}(\hat{c}_{\psi(v)}) \right), \tag{11}$$

where $\psi(v)$ maps v to its segment, and the maximum is taken over all topology intervals in which v appears.

4) Temporal Aggregation and Scoring: The graph-level readouts from N_T topology intervals form a sequence $\{r^{(1)}, \ldots, r^{(N_T)}\}$ that is fed into a two-layer LSTM. A linear projection with softmax over the LSTM output yields a scalar graph-level probability $p_{\mathcal{G}}$. The root cause probability of node v combines the graph-level and node-level signals:

$$R_v = p_{\mathcal{G}} \cdot a_v. \tag{12}$$

For nodes appearing in multiple intervals, probabilities are accumulated across occurrences.

5) Training Objective: We train HCSTGAN with a back-propagation loss that encodes the causal direction of failure propagation. For each anomalous node $\alpha \in \mathcal{A}$, the target probability is 1. For each predecessor β of α, a learnable weight $w_{\alpha\beta}$ quantifies the propagation strength:

$$\mathcal{L}_{bp} = \sum_{\alpha \in \mathcal{A}} \left[(1 - R_\alpha)^2 + \sum_{\beta \in \text{Pred}(\alpha)} (w_{\alpha\beta} - R_\beta)^2 \right], \tag{13}$$

where $\text{Pred}(\alpha)$ is the predecessor set of α. The weights $w_{\alpha\beta}$ are initialized to 1 and jointly learned, converging to values that reflect causal strength. This enables HCSTGAN to identify hidden root causes whose own metrics do not fluctuate, since their influence is encoded in the weights leading to downstream anomalous nodes.

C. Confidence-Gated Agent Reranking

While HCSTGAN delivers high accuracy in the majority of cases, two failure modes warrant additional reasoning beyond pure neural scoring. The first is when the top scores are close and the network cannot separate candidates decisively. The second is when the top-ranked node is a gateway service, which absorbs symptoms without being the cause. We address both through a gated agent invoked selectively.

1) Confidence-Based Triggering: Let $\{(v_1, R_{v_1}), (v_2, R_{v_2}), \ldots\}$ be the ranking sorted by HCSTGAN scores. The confidence gap is

$$\Delta_{\text{conf}} = R_{v_1} - R_{v_2}. \tag{14}$$

The agent is activated when $\Delta_{\text{conf}} < \tau_{\text{conf}}$, or when v_1 matches a known victim pattern such as a gateway service that often absorbs upstream symptoms. Otherwise the HCSTGAN ranking is returned directly.

2) Structured Prompt and Context: When triggered, HAGR extends the candidate set to the top-K_s nodes and sends their context to the agent. For each candidate, the context includes the normalized HCSTGAN score, resource metrics (CPU and memory utilization), service-level indicators (success rate and latency), and downstream dependencies from the topology. The prompt follows the template shown in Fig. 3, in which metric names are abstracted into slots so the same template applies across deployment scenarios. The template encodes three reasoning steps: check resource saturation, distinguish culprit from victim, and integrate the algorithmic prior. The agent returns a refined ranking in JSON format.

Prompt Template

[SYSTEM ROLE]
You are an expert SRE. Rerank the candidates to identify the root cause based on the provided logic.

[METRIC SCHEMA]
- `<RESOURCE>`: Primary bottleneck indicator.
- `<HEALTH>`: Service level indicator.
- `<PRIOR>`: Algorithmic suspicion score.

[INPUT DATA]
`Context = {{SERIALIZED_CANDIDATES}}`

[REASONING LOGIC]
1) **Check Resource Saturation**
   - *Condition:* If `<RESOURCE>` $> \tau_{sat}$.
   - *Action:* Mark as high confidence root cause.
2) **Distinguish Symptoms**
   - *Culprit:* Poor `<HEALTH>` AND high `<RESOURCE>`.
   - *Victim:* Poor `<HEALTH>` BUT low `<RESOURCE>`.
3) **Integrate Algorithmic Prior**
   - Trust `<PRIOR>` unless contradicted by Step 1.

[OUTPUT FORMAT]
Return JSON: `{"ranking": ["ID_1", "ID_2", ...]}`

Fig. 3: The structured reasoning prompt template. Specific metric names are abstracted into slots to generalize the diagnosis logic across deployment scenarios.

3) Neurosymbolic Score Fusion: The final ranking fuses HCSTGAN scores with the agent output. Let $\bar{R}_v = (R_v - R_{\min})/(R_{\max} - R_{\min})$ be the min-max normalized HCSTGAN score, and $S_v^{\text{Agent}} = 1/\text{rank}_{\text{Agent}}(v)$ be the inverse-rank score from the agent:

$$R_v^{\text{fused}} = \alpha_{\text{GNN}} \cdot \bar{R}_v + \beta_{\text{Agent}} \cdot S_v^{\text{Agent}}, \tag{15}$$

where $\alpha_{\text{GNN}} + \beta_{\text{Agent}} = 1$.

4) Gateway Suppression with Safety Guards: Gateway services may still receive high fused scores, since their resource metrics can stay normal while their traffic-level indicators look anomalous. We add a symbolic rule that activates when two conditions hold together. The top-ranked candidate after fusion is a gateway-type node with healthy resource utilization, and another candidate in the top-K_s set shows clear resource saturation. When both hold, a suppression factor $\eta < 1$ is applied:

$$R_{\text{gateway}}^{\text{fused}} \leftarrow \eta \cdot R_{\text{gateway}}^{\text{fused}}. \tag{16}$$

The above fusion is further protected by a safety guard that arbitrates between the two modules when they disagree. Whenever the agent promotes a node from a known noise category that HCSTGAN does not flag, the agent output is discarded and the HCSTGAN ranking is retained. This design ensures that symbolic reasoning refines neural evidence only when itself trustworthy, completing HAGR as a robust pipeline.

IV. Experiments

A. Experimental Setup

1) Datasets: We use two hybrid-deployed cloud-edge benchmarks released by Zhu et al. [7], denoted as BH and SH.

TABLE I: Statistics of the two datasets.

Dataset	Root-cause system	Service classes	Metric samples
BH	Bookinfo	36	3,474,000
SH	SockShop	84	8,106,000

Each benchmark co-deploys four open-source microservice systems on a Kubernetes cluster spanning one cloud node and two edge nodes. The dataset name encodes the system in which the root cause is located. Table I shows the details of each dataset.

2) Baselines: We compare HAGR with following methods. CloudRanger [3] builds a service causal graph using the PC algorithm on Pearson correlations and performs second-order random walk for ranking. MicroCERCL [7] is the strongest published method on these benchmarks. It trains a graph neural network over a heterogeneous topology stack with a backpropagation strength loss. RCLAgent [11] drives an LLM agent with structured prompts and tool-augmented reasoning over service traces and metrics.

3) Evaluation Metrics: We report ACC@K for $K \in \{1, 3, 5\}$ and mean reciprocal rank (MRR). ACC@K is the proportion of cases where the true root cause appears in the top K. MRR is defined as $\mathrm{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\mathrm{rank}_i}$, where rank_i is the position of the true root cause in the i-th case.

4) Implementation: We conduct all experiments with Python 3.7, PyTorch 1.13.1, DGL 0.9.1, and geoopt 0.5.0. All experiments run on a Linux workstation with an Intel Xeon Gold 6226R 2.90GHz CPU, 256GB RAM, and an NVIDIA RTX 3090 GPU. HCSTGAN uses a hidden dimension of 64, four attention heads, and curvature $c = 1$. The Birch threshold of the upstream anomaly detector is 0.07. The confidence threshold $\tau_{\text{conf}} = 0.05$, candidate set size $K_s = 15$, gateway suppression factor $\eta = 0.5$, and fusion weights $\alpha_{\text{GNN}} = 0.6$, $\beta_{\text{Agent}} = 0.4$. The agent is built on deepseek-chat.

B. Comparison with Baselines

Fig. 4 and Fig. 5 summarize the comparison results from two observations. HAGR achieves better than other models. On ACC@1, HAGR improves over MicroCERCL by 16.3% and 7.2% on BH and SH. On MRR, HAGR reaches 0.677 and 0.789, raising MicroCERCL by 0.111 and 0.064.

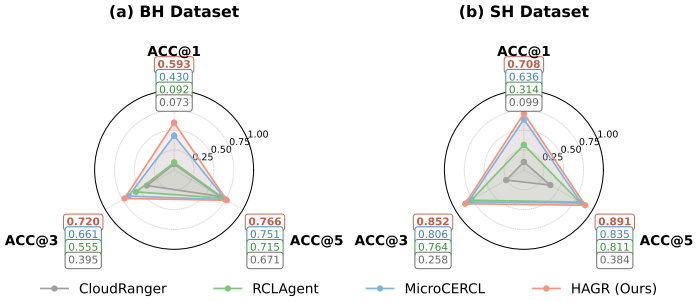

Fig. 4: ACC@K comparison between HAGR and baselines. HAGR is the outermost polygon on every dataset.

CloudRanger relies on Pearson correlation and random walk, which assume stable service neighborhoods. Cross-segment latency variance under hybrid deployment violates this assumption, so its ACC@1 stays below 0.10 on both datasets. RCLAgent reaches reasonable ACC@3 but its ACC@1 stays below 0.34 on SH. The agent reasons from raw metrics without a structured prior, and gateway services with amplified symptoms are easily promoted to the top. MicroCERCL is the strongest baseline because it learns failure backpropagation with a graph neural network. However, it embeds topology in Euclidean space, which cannot represent the exponential branching of cloud-edge hierarchies without distortion. The gap is largest on BH, where high-degree services within the same network segment create deeper structure.

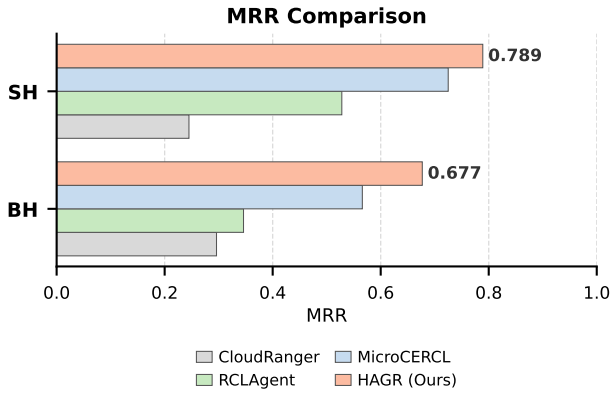

Fig. 5: MRR comparison on two datasets.

HAGR addresses both limitations in a complementary manner. The hyperbolic graph attention preserves hierarchical relations and widens the score margin among candidates, while the confidence-gated agent resolves the residual cases in which the GNN remains indecisive. The gain on ACC@5 is smaller because all top methods already cover most true root causes within the first five positions.

C. Ablation Study

We construct four variants to disentangle the contribution of each component. Base disables both contributions and uses Euclidean graph attention with no agent. w/o Agent keeps HCSTGAN but skips agent reranking. w/o Hyp keeps the agent but uses Euclidean attention. HAGR (Full) enables both. All variants share the same anomaly detector, training schedule, and hyperparameters. Table II reports absolute scores. Fig. 6 shows the relative improvement of HAGR (Full) over each variant.

1) Effect of HCSTGAN: Comparing Base with w/o Agent isolates the contribution of HCSTGAN. ACC@1 rises from 0.461 to 0.502 on BH and from 0.598 to 0.658 on SH. The Poincaré ball bounds the embedding norm and forces hierarchically distant nodes apart, which sharpens the score gap. The network-segment attention pooling aggregates evidence

within a segment before propagating it across segments, which matches the regional concentration of cloud-edge deployments and reduces cross-segment noise.

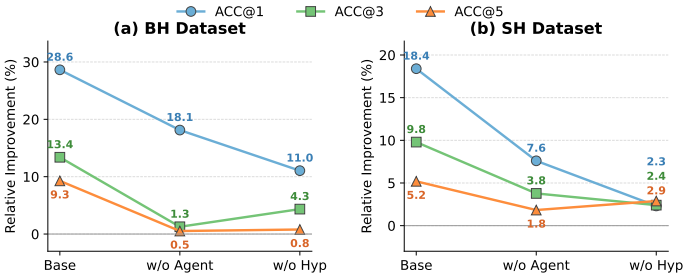

Fig. 6: Relative improvement of HAGR (Full) over each ablated variant on ACC@1, ACC@3, and ACC@5.

2) Effect of agent reranking: Comparing Base with w/o Hyp measures the standalone contribution of the agent. ACC@1 improves from 0.461 to 0.534 on BH and from 0.598 to 0.692 on SH. The gain on ACC@1 is consistently larger than on ACC@5, since the agent is invoked when the top-2 score gap is small and its corrections concentrate on the head of the ranking. The gateway suppression rule also shows its effect on both datasets, where many false alarms in Base land on Bookinfo's productpage and SockShop's front-end, both of which are gateway services in their respective systems.

3) Synergy: Fig. 6 reveals two consistent patterns across both datasets. The ACC@1 line stays above ACC@3 and ACC@5 on both datasets, which means both components target ranking precision at the top rather than recall. The improvement of HAGR over Base reaches 28.6% and 18.4% on BH and SH, exceeding the sum of its improvements over w/o Agent and w/o Hyp on both datasets. This indicates that HCSTGAN and the agent address complementary failure modes. The largest gain occurs on BH, where high-degree services concentrated in the same network segment give the hierarchical and gateway-suppression mechanisms the most room to act.

V. Conclusion

In this paper, we proposed a new framework for root cause localization in hybrid-deployed cloud-edge microservice systems called HAGR. Our framework addressed hierarchical distortion through a hyperbolic constrained spatio-temporal graph attention network, which preserved tree-like service dependencies on the Poincaré ball and aggregated evidence at the network-segment level. We introduced confidence-gated agent reranking to selectively invoke language-model reasoning when the top-two score margin is small, and employed a symbolic gateway-suppression rule to correct residual false alarms on traffic-aggregating services. The results showed that our method has significantly outperformed existing methods, improving ACC@1 by 16.3% and 7.2% over the strongest baseline on two cloud-edge benchmarks. Future work will focus on online adaptation to topology drift and integrating its hyperbolic reasoning principles with lightweight distilled agents.

TABLE II: Ablation results on BH and SH. The best result in each column is in bold.

Dataset	Variant	ACC@1	ACC@3	ACC@5	MRR
BH	Base	0.461	0.635	0.701	0.580
	w/o Agent	0.502	0.711	0.762	0.624
	w/o Hyp	0.534	0.690	0.760	0.638
	HAGR (Full)	**0.593**	**0.720**	**0.766**	**0.677**
SH	Base	0.598	0.776	0.847	0.710
	w/o Agent	0.658	0.821	0.875	0.755
	w/o Hyp	0.692	0.832	0.866	0.770
	HAGR (Full)	**0.708**	**0.852**	**0.891**	**0.789**

Acknowledgment

This work was supported in part by the Leading Talent Program for Industrial Science and Technology Innovation of Jiangsu Province under Grant No. 74032505, and in part by the Start-up Funding for 2025 National High-level Talent Frontier Introduction Project (01) under Grant No. 70012501A01.

References

[1] K. Fu, W. Zhang, Q. Chen, D. Zeng, and M. Guo, "Adaptive resource efficient microservice deployment in cloud-edge continuum," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1825–1840, 2021.

[2] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, W. Li, and D. Ding, "Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study," *IEEE Transactions on Software Engineering*, vol. 47, no. 2, pp. 243–260, 2018.

[3] P. Wang, J. Xu, M. Ma, W. Lin, D. Pan, Y. Wang, and P. Chen, "Cloudranger: Root cause identification for cloud native systems," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2018, pp. 492–502.

[4] L. Wu, J. Tordsson, E. Elmroth, and O. Kao, "Microrca: Root cause localization of performance issues in microservices," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2020.

[5] S. Chakraborty, S. Garg, S. Agarwal, A. Chauhan, and S. K. Saini, "Causil: Causal graph for instance level microservice data," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 2905–2915.

[6] R. Xin, P. Chen, and Z. Zhao, "Causalrca: Causal inference based precise fine-grained root cause localization for microservice applications," *Journal of Systems and Software*, vol. 203, p. 111724, 2023.

[7] Y. Zhu, J. Wang, B. Li, X. Tang, H. Li, N. Zhang, and Y. Zhao, "Root cause localization for microservice systems in cloud-edge collaborative environments," *arXiv preprint arXiv:2406.13604*, 2024.

[8] E. Chien, C. Pan, P. Tabaghi, and O. Milenkovic, "Highly scalable and provably accurate classification in poincaré balls," in *2021 IEEE International Conference on Data Mining (ICDM)*, 2021, pp. 61–70.

[9] Z. Wang, Z. Liu, Y. Zhang, A. Zhong, J. Wang, F. Yin, L. Fan, L. Wu, and Q. 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*, 2024, pp. 4966–4974.

[10] X. Zhou, G. Li, Z. Sun, Z. Liu, W. Chen, J. Wu, J. Liu, R. Feng, and G. Zeng, "D-bot: Database diagnosis system using large language models," *arXiv preprint arXiv:2312.01454*, 2023.

[11] L. Zhang, T. Jia, K. Wang, W. Hong, C. Duan, M. He, and Y. Li, "Adaptive root cause localization for microservice systems with multi-agent recursion-of-thought," *arXiv preprint arXiv:2508.20370*, 2025.

[12] B. Lorbeer, A. Kosareva, B. Deva, D. Softić, P. Ruppel, and A. Küpper, "Variations on the clustering algorithm birch," *Big data research*, vol. 11, pp. 44–53, 2018.