Subgraphs as First-Class Citizens in Incident Management for Large-Scale Online Systems: An Evolution-Aware Framework

Zilong He, Pengfei Chen*, Yu Luo, Qiuyu Yan, Hongyang Chen, Guangba Yu, Fangyuan Li, Xiaoyun Li, and Zibin Zheng, *Fellow, IEEE*

***Abstract*—With the ever-increasing scale and complexity of modern online systems, incidents are becoming inevitable, which seriously decreases the system availability and user satisfaction. To enhance incident management, many machine learning based techniques are proposed to automate incident detection and diagnosis. However, previous studies have mostly ignored the impact of evolution on the practicality of an incident management framework. Specifically, (1) The scale of modern online systems is continually evolving, but most state-of-the-art techniques are overly dependent on a continuous modelling of the entire system, and thus are less practical for online systems evolved to tens of thousands of services; (2) The volume of telemetry data is massively growing, while the number of incident records for learning is scarce and slowly generated (sometimes from zero), but prior techniques usually neglect this extreme imbalance in data volume evolution, and cannot support the life-cycle evolution (i.e., cold start and continual learning) of their developed models; (3) Prior techniques usually require operators to manually select a set of telemetry as inputs for incident diagnosis, but ignore how to automatically evolve this selection to continually improve diagnosis performance. These gaps stem from the unawareness of evolution, including the evolution of the target online system and the evolution of the built incident management models. To fill these gaps, we propose an evolution-aware incident management framework GEM. Specifically, considering the evolution of system scale and data volume, GEM continually refines the enormous real-time collected telemetry data into individual compact yet expressive graph-based representations, namely issue impact subgraphs, and treat them as the first-class citizens in incident management. Centered around these subgraphs, we design a couple of lifelong learning based graph analysis techniques to learn and evolve models for incident detection and diagnosis. We evaluate GEM using real-world data collected from the WeChat online system, the largest instant messaging software in China. The results confirm the effectiveness of GEM. Moreover, GEM is successfully deployed in WeChat, easing the burden of operators in handling a flood of issues and related telemetry data.**

***Index Terms*—Software Operation, Incident Management, Online Systems, Graph Learning, Continual Learning**

I. INTRODUCTION

ONLINE systems are software organized to deliver diverse functions through inter-dependent services, and are generally accessible over the internet. By serving users through web browsers, mobile apps, or other internet-connected interfaces, these systems have been pervasive in our everyday activities, such as communication, information sharing and entertainment. With the rapid evolution of functionalities and the growth of users, the scale and complexity of these systems are becoming larger and larger. Such online systems are built upon numerous services implemented with a multitude of software technologies to support an abundance of requirements. Therefore, their operation and maintenance are crucial software engineering activities, yet are extremely difficult. Despite a quantity of effort devoted to service availability assurance, availability incidents, which impact system uptime and accessibility, are still inevitable. Unlike security incidents that involve data leakage or privacy violations, availability incidents directly affect the ability of users to access services. This study focuses specifically on availability incidents, and for simplicity, the remainder of this paper will use the term incident to exclusively denote availability incidents. Without timely and appropriate management, such incidents can quickly result in great economic loss and a serious decrease of user experience. For example, it is estimated that Amazon.com has lost over $100 million for a single hour of downtime on Prime Day in 2018 [1]. Therefore, it is a critical task to manage incidents efficiently and properly.

Incident management follows a lifecycle from detection through diagnosis to final mitigation actions. In this paper, we denote issues as aggregations of system anomalies identified from telemetry data. Then, incidents—the focus of this study—are issues that require immediate attention. During the detection phrase, identifying true incidents among numerous potential issues presents the first challenge. Once detected, operators must diagnose root causes, which then guides efficient mitigation. These tasks are challenging as they require integrating diverse information to make accurate judgments, including what are affected (e.g., the attributes of affected services) and how they are affected (i.e., issue symptoms described with the telemetry data).

Recently, tremendous efforts are devoted to automate incident detection [2]–[6] and diagnosis [7]–[18]. The basic idea of most state-of-the-art methods is to formulate the problems as a classical machine learning problem, e.g., classification. To this end, a dataset associated with a sufficient number of incident records and an abundance of continuously-collected telemetry data, is used for model learning. Their ultimate goal is to propose a more powerful model (e.g., neural networks)

Zilong He, Hongyang Chen, Guangba Yu, Xiaoyun Li and Pengfei Chen are with the School of Computer Science and Engineering, Sun Yat-sen University, China. Zibin Zheng is with the School of Software Engineering, Sun Yat-sen University, China. E-mail: {hezlong, chenhy95, yugb5, lixy223}@mail2.sysu.edu.cn and {chenpf7, zhzibin}@mail.sysu.edu.cn.

Yu Luo, Qiuyu Yan and Fangyuan Li are with Tencent Inc., China. E-mail: {zekaluo, ireneyan, leiffyli}@tencent.com.

Pengfei Chen is the corresponding author.

that can better fit the data for incident management.

While machine learning techniques have demonstrated increasing accuracy in incident management, these approaches often do not account for system evolution, which is common in practice and can significantly impact framework practicality across multiple dimensions. Specifically, this paper studies incident management under the evolution of system scale, data volume and diagnosis telemetry.

The Evolution of System Scale. Online systems are continuously scaling and evolving. However, many prior methods for incident detection or diagnosis [5]–[17] assume a small scale or static topology of the target system, and then directly conduct whole-system modelling, making them inefficient and impractical to scale out to large-scale and continually-evolving online systems. Generally, we consider an online system with a large number of services (e.g., more than 1K services) as large-scale. Compared with prior studies, this study targets at building an incident management framework that is intrinsically evolvable to the system scale.

The Evolution of Data Volume. The volume of telemetry data collected in a large-scale online system is usually overwhelming and continuously growing. Many organizations apply data retention policies (e.g., deleting telemetry data routinely) to reduce storage cost, which hinders the accumulation of incident records and related data. Although recent data sketching techniques [19] or CPU-and-memory-constrained anomaly detection methods [20] can help reduce the monitoring overhead and storage cost, they cannot facilitate the accumulation of expressive incident records needed for the model learning in diverse incident management tasks. This scarcity of expressive incident records is a main factor preventing prior incident management methods from being production-ready for modern large-scale online systems. Specifically, prior methods fall into two categories: supervised learning-based and heuristic-based methods. Supervised learning-based methods require extensive historical telemetry data and abundant incident records for analysis [2]–[5], [7]–[11], while heuristic-based methods rely on manually generated heuristics to perform incident management [6], [12]–[18]. Consequently, supervised learning-based methods are sometimes impractical due to their stringent requirements, while heuristic-based approaches suffer from the failure to automatically incorporate the knowledge in historical incident records. Compared with prior studies, our study avoids relying on continuous whole-system telemetry data persistence, and targets at building an intelligent incident management framework that can kick-start incident management when incident records are few, and then gradually refine itself with continual learning.

The Evolution of Diagnosis Telemetry. The telemetry considered useful for incident diagnosis (denoted as diagnosis telemetry in the following) needs to evolve as more incidents have been analyzed. While prior studies [7]–[18] often rely on pre-determined sets of diagnosis telemetry, which are manually summarized from established ICT system design practices, we hypothesize that comprehensive and explainable incident diagnosis models can be automatically built from observational data without relying heavily on a priori system design or behavior knowledge. To this end, this study proposes a framework that automatically expands the set of telemetry for incident diagnosis, complementing existing approaches that are built upon blueprints with indications of the diagnosis telemetry and incident modes of the systems.

Faced with these problems, we propose a framework GEM for subGraph-centered Evolution-aware incident Management in large-scale online systems. Here, the term "evolution-aware" comprises two levels of meanings. From the macro level, we take account of the three system-and-framework evolution problems discussed above to make the evolution (i.e., continual data preparation and model training) of the overall framework more smooth. This smoothness is achieved through the incorporation of cold start mechanisms that enable the framework to function effectively with limited initial data, as well as continual learning techniques that allow the framework to adapt to system evolution. From the micro level, when representing each specific issue for incident management, we consider how it evolves from a fault in the root cause service to a bundle of affected services. Then, the term "subgraph" entails one of our key ideas, which will be elaborated below.

Specifically, facing the evolution of system scale and data volume, we argue that some compact but yet expressive issue representations should be refined from the telemetry data as early as possible. To this end, before incident detection and diagnosis, we use graphs to organize information describing the impact scope of issues. Each issue is then represented as a subgraph of the system topology associated with related impact information for incident management. The scales of subgraphs are much smaller than the whole-system graph. We model such issue impact subgraphs as the first-class citizens in incident management. Specifically, first-class citizens denote elements that can be dynamically created, passed between different procedures, and persisted for on-demand analysis. Our approach differs from prior data-driven methods by putting forward impact extraction in front of other incident management tasks, enabling us to model impact subgraphs as first-class citizens in incident management. In contrast, prior data-driven approaches [2]–[18] usually treat the whole-system snapshots as first-class citizens for incident management. While some of them [5]–[18], [21], [22] have also utilized graph-based approaches to circumscribe the cause-effect search space, they typically model whole-system graphs, which becomes impractical for large-scale online systems. In contrast, we extract, persist, and model each incident as a system subgraph, providing a more focused and scalable representation.

Centered around the impact subgraphs, we build downstream models that can well handle diverse information to automate incident management. Specifically, for incident detection, we design an edge conditioned graph neural networks based model to comprehensively incorporate different sources of information. Then for incident diagnosis, we design an evolutionary PageRank based mechanism that can learn to well balance diverse clues with the consideration of their causal relationships and strengths. Moreover, we design lifelong learning mechanisms (i.e., cold start and continual learning) for the evolution of all the proposed models. Extensive evaluations show that our GEM framework outperforms existing methods in terms of incident detection, improving F1-score

by 150%∼270%, and incident diagnosis, improving Top-1 Accuracy by 43.45%∼159.54%.

To sum up, the contributions of this paper are as follows.

- We summarize key challenges of incident management in practice from the perspective of evolution, including the evolution of system scale, data volume and diagnosis telemetry. Moreover, we reformulate the overall incident management problem as a subgraph-centered lifelong learning problem, which demonstrates great practicality and effectiveness for incident management in large-scale online systems.
- We design effective techniques to perform incident detection and diagnosis based on the extracted impact subgraphs, which can effectively incorporate diverse incident related information, including the system topology, affected service attributes and issue symptoms.
- We conduct experiments to evaluate our framework based on open-sourced data and real-world data. We also integrate it with the incident management in the WeChat online system, a real-world large-scale online system supporting billions of users. The results confirm the effectiveness of our framework. The artifact is available at https://github.com/IntelligentDDS/GEM.

II. Motivations and Problem Formulation

A. State-of-the-Art Methods on Incident Management

This paper primarily focuses on three key problems in incident management including impact extraction, incident detection, and incident diagnosis. We first introduce how current works [2]–[18], [23]–[26] formulate these problems in this section, and then in the next three sections, we will analyze their limitations (Section II-B), discuss our key ideas (Section II-C), and refine the problem formulation (Section II-D).

Impact Extraction. This process correlates and aggregates abnormal telemetry data into disjoint subsets, grouping those with the same root cause. Each subset forms an impact subgraph $T = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E}, \mathbf{G})$, where $\mathcal{V}$ represents affected nodes (e.g., services), $\mathcal{E}$ represents abnormal relationships between nodes, $\mathbf{X}$ denotes node features, $\mathbf{E}$ represents edge features, and $\mathbf{G}$ captures global subgraph features.

Incident Detection. Given telemetry data $\mathcal{W}_{\widehat{\mathcal{V}}} = \{\mathbf{W}_i | i \in \widehat{\mathcal{V}}\}$ from all monitored entities $\widehat{\mathcal{V}}$, this process determines whether an incident occurs, represented as $y \in \{0, 1\}$, where 1 indicates an incident, i.e., an emergent issue.

Incident Diagnosis. Given telemetry data $\mathcal{W}_{\widehat{\mathcal{V}}}$ and system topology $\widehat{\mathcal{E}}$, this process aims to identify the root cause $r \in \widehat{\mathcal{V}}$.

Our Approach vs. Prior Methods. Existing methods typically treat impact extraction [23]–[26] and incident detection/diagnosis [2]–[17] as separate processes, or only incorporates impact extraction during diagnosis [18]. We propose positioning impact extraction before incident detection and leveraging its outputs throughout the incident management pipeline. Our motivation and refined problem formulation are discussed in the following sections.

B. Over-optimistic Assumptions in Prior Works

The goal of our study is to build an incident management framework that can counter some over-optimistic assumptions in prior works, which are detailed below.

Over-optimistic Assumption #1: The target online system will always be small and will not evolve. Most of prior works [5]–[17] perform whole-system modelling during building models for incident management. Specifically, they model the recorded whole-system snapshots, which contain all the telemetry data from all the entities of a system. When the target usage scenario is an evolution-free online system with only tens of services, this is plausible. However, real-world online systems are rapidly evolving, and some of them may eventually become large-scale. For example, the number of services of Wechat backend system has evolved from around 3K [27] to over 20k now. Specifically, the number of services that are part of this study is 21871, and the number of the call-relationships between them is 85966. For such a continually-evolving and large-scale online system, a whole-system modelling method may be unavailable when the system topology has evolved, or when organizations cannot afford the persistence of all historical whole-system snapshots. For example, deleting outdated telemetry data routinely is a regular practice to save operation cost. Therefore, compared with prior works, GEM builds models which naturally operate on individual local views of an online system, thereby avoiding excessive processing and endless reliance on whole-system telemetry data.

Over-optimistic Assumption #2: A dataset with abundant incident records and related information is always ready for analyzing the rules behind incidents. Historical incident information are essential for forming incident management solutions. Specifically, for supervised learning based techniques [2]–[5], [7]–[11], the parameters of a data-driven model are explicitly learned from labelled incidents. Even for heuristic based and unsupervised learning based methods [6], [12]–[18] which do not learn from labels explicitly, their authors have to inspect historical incidents to form well-grounded heuristics. As a result, prior works generally assume that the whole-system telemetry data can be continuously persisted to prepare for a comprehensive dataset, and considers and evaluates the effectiveness of an incident management model after it has been properly trained or set up using this comprehensive dataset. However, in practice, comprehensive incident datasets containing all necessary information for model training are rarely available, and in some cases, datasets may be entirely absent initially, necessitating collection efforts to begin from scratch. This is because incident records are usually scarce and generated slowly (e.g., an incidence rate of lower than 2.5×10^{-4} incidents per service per day according to our collected data from WeChat), while telemetry data in large-scale online systems are of great volume (e.g., only call-relationship level telemetry data take up over 500TB per day in WeChat) and thus usually cannot be persistent forever for any-time analysis. Such a dilemma calls for reformulations of the incident management problems to avoid the strict reliance on continuous whole-system telemetry data persistence, and a

new incident management model that can cold start and then be incrementally learned from a continually expanding set of incident records.

Over-optimistic Assumption #3: All useful telemetry can be determined in advance for incident diagnosis. Prior works for incident diagnosis [7]–[18] usually assume that all useful telemetry have been manually picked out and the only task is to build a powerful model accordingly. Such an assumption poses a great pressure to operators. Specifically, as pointed out by the common machine learning principle "Garbage in, Garbage out", the quality of telemetry selected by operators will become a key bottleneck of the performance of an incident diagnosis algorithm. If operators cannot select most of the useful telemetry and remove most of the noisy, redundant or useless ones at the beginning, the performance of an incident diagnosis algorithm will be hard to guarantee. However, selecting useful telemetry in advance is not an easy task since the distribution of possible faults may evolve and usually needs to be gradually unveiled. Therefore, compared with prior works, GEM intends to build an incident diagnosis method that can automatically identify useful telemetry over time, and then gradually take them into consideration via online lifelong learning.

C. Our Key Ideas

Our proposed evolution-aware incident management framework mainly comprises the following two key ideas.

Representing and Persisting Issues/Incidents as Impact Subgraphs instead of Whole-System Snapshots. To cope with the evolution of system scale and data volume, we extract impact subgraphs to represent issues, and treat them as first-class citizens in incident management. Compared with whole-system snapshots, subgraphs are more expressive because irrelevant information are filtered out and thus do not need to be repeatedly processed. Besides, in the context of large-scale online systems with tens of thousands of services, persisting subgraphs is more economical than persisting all the telemetry data all the time. For example, in WeChat, the raw telemetry data for call-relationships can take up over 500TB per day, while the impact subgraphs that need to be persisted for model learning and evaluation take up only about 10MB per day (details about the extraction and persistence process are in Section III-B). In general, subgraph based issue representations are more suitable for incident management knowledge accumulation when the target online system may evolve to large-scale.

Learning Detection and Diagnosis Models in a Manner of Lifelong Learning instead of Fixed-Set Learning. To cope with the evolution of data volume and diagnosis telemetry, we consider the full life-cycle evolution of the incident management framework itself, including how to resolve the cold-start problem when only very few historical incident records are available, and how to continually optimize detection and diagnosis models after they have been deployed. Compared with fixed-set learning which assumes sufficient training data and an unchanged set of considered features, a lifelong learning driven solution can better match the practice.

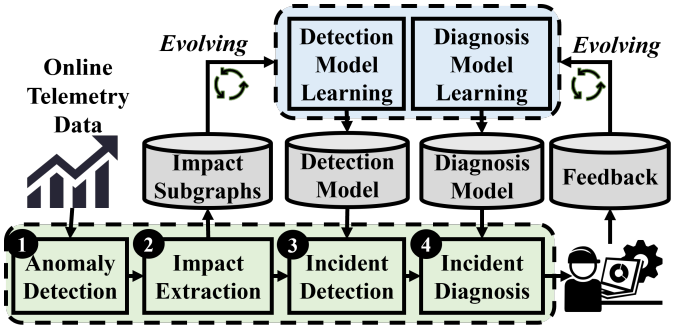


Fig. 1. The overview of GEM.

D. Refined Problem Formulation

Based on the above ideas, the problem formulation can be further refined as follows. It is noted that the definition of an impact subgraph $T = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E}, \mathbf{G})$ follows the formulation in Section I.

Incident Detection with Subgraphs as First-Class Citizens. Given an impact subgraph $T = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E}, \mathbf{G})$, the objective of subgraph-centered incident detection is to determine whether T implies an incident, and finally output its associated detection result $y \in \{0, 1\}$, where 1 represents T is the impact subgraph of an incident, and 0 otherwise.

Incident Diagnosis with Subgraphs as First-Class Citizens. Given an incident whose impact subgraph is $T = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E}, \mathbf{G})$, the objective of subgraph-centered incident diagnosis is to locate its root cause entity $r \in \mathcal{V}$, which is the culprit of the incident.

Model Evolution with Lifelong Learning. This study considers the evolution perspective during solving the above problems. Specifically, the set of subgraph based incident records $\{(T_t, y_t, r_t)|t = 1, 2, \cdots\}$ for model learning or rule formulation is gradually expanded from $\emptyset$, and the related models need to be incrementally extended and refined in such a situation. This problem, especially how to cold start, is rarely discussed in prior works on incident management, but has a practical value in the real-world deployment of incident management algorithms.

III. THE PROPOSED APPROACH

In this study, we propose **GEM**, a novel evolution-aware framework for incident management in large-scale online systems. Fig. 1 presents the overall workflow of GEM. The online deployment of GEM (highlighted in the green box) consists of four major phases, namely ❶ anomaly detection, ❷ impact extraction, ❸ incident detection and ❹ incident diagnosis. Among these processes, ❸ incident detection and ❹ incident diagnosis are supported by the models learned by GEM (highlighted in the blue box). These models will gradually evolve as more data and feedback are collected. In the following, we will elaborate these processes in detail.

Working Hypothesis. To work effectively, GEM requires several key information sources including service attribute information stored in CMDB, service call-relationship topology, telemetry data collected from the system, and operator feedback on detection results and diagnosed root causes. If some

information is missing, organizations need to upgrade their monitoring infrastructure to collect these data, for example, by implementing service mesh solutions like Istio [28] to capture service interactions and telemetry.

A. Anomaly Detection

The first step is anomaly detection. Specifically, abnormal entities and relationships are figured out first in real time, acting as an entry point of subsequent phases. To this end, we employ a day-by-day difference based method to detect anomalies from the real-time telemetry data of entities (e.g., services) and relationships (e.g., call-relationships) from all over the system. Here, day-by-day difference is defined as the absolute difference between the current value of an observation and its value at the same time in the previous day. Specifically, for an observation x at time t (here, x can be $\mathbf{X}_{i,k}$ for telemetry k on entity i, or $\mathbf{E}_{(i,j),k}$ for telemetry k on relationship (i,j)), its day-by-day difference $\mathrm{Diff}(t)$ is:

$$\mathrm{Diff}(t) = |x(t) - x(t - 24h)|, \tag{1}$$

where $x(t)$ represents the current value of the observation and $x(t-24h)$ represents the value at the same time in the previous day. For each telemetry, we establish a baseline distribution of normal behavior by collecting these differences from historical data during normal operation periods. From this distribution, we compute a threshold θ as the 99th percentile:

$$\theta = \mathrm{Percentile}_{99}(\mathrm{Diff}(t) | t \in \text{normal operation periods}) \tag{2}$$

For each telemetry, an observation is flagged as anomalous when its day-by-day difference exceeds this threshold. One can also replace this anomaly detection algorithm with anomaly detectors [29], [30] developed in recent years.

B. Impact Extraction

Detected anomalies need to be aggregated into individual issue impact subgraphs for accurate analysis, especially when multiple issues co-exist in large-scale systems. GEM extracts these subgraphs based on relationship-level call Failure Counts per minute (FC). A call failure occurs when a service cannot receive a correct response from another service (e.g., a non-zero status code in gRPC). Issues typically manifest as call failures across services—whether from software bugs that crash services or performance issues that trigger timeout mechanisms. Consequently, FC usually serves as a Key Performance Indicator (KPI) or golden metric [31].

Inspired by related works [16], [32], we consider two observed FC windows to share the same root cause when they: (i) exhibit similar abnormal temporal patterns over the current time window, and (ii) belong to the same connected component in the graph of detected abnormal call-relationships.

To satisfy condition (i), we perform shape-based clustering on FC windows. For two abnormal FC windows, $\mathbf{W}_{(i,j),k}$ and $\mathbf{W}_{(p,q),k}$ (where k denotes FC and (i,j) and (p,q) are call-relationships), we calculate their shape-based distance as:

$$\mathrm{Dist}_k\Big((i,j),(p,q)\Big) = \begin{cases} 1 & \text{if } |\rho(\mathbf{W}_{(i,j),k}, \mathbf{W}_{(p,q),k})| \geq 0.9 \\ 2 & \text{otherwise} \end{cases}. \tag{3}$$

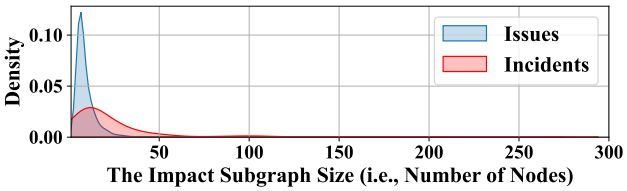


Fig. 2. The kernel density estimate plot for the extracted impact subgraph size of issues and incidents in WeChat over one month. The impact subgraph size distribution follows a long-tail distribution, with the largest value as 268.

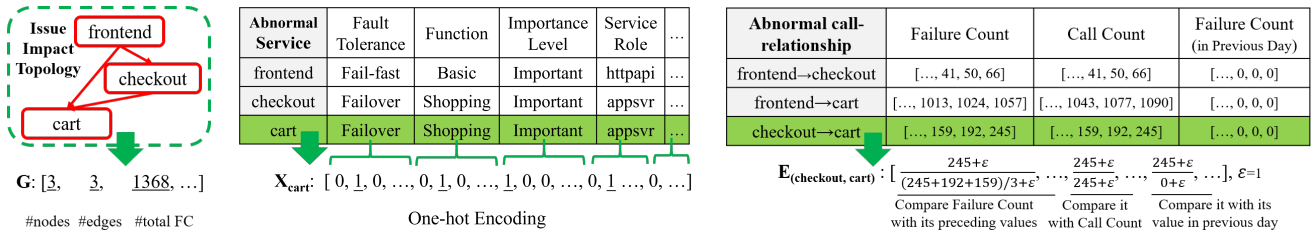
Specifically, when the absolute Pearson Correlation between two FC windows exceeds 0.9, we consider them similar in shape and assign a smaller distance value. Using this distance measure, we apply DBSCAN [33] to cluster FC windows. DBSCAN is chosen because it does not require pre-specifying the number of clusters. To meet condition (ii), we extract connected components from each cluster to form individual impact subgraphs. By doing so, the extracted issue impact subgraphs are connected graphs with call-relationships showing similar abnormal patterns.

GEM performs this impact extraction procedure every minute, linking subgraphs across adjacent minutes if they share the same automated root cause (detailed in Section III-D). Small (fewer than 3 nodes) or transient (less than 3 minutes) subgraphs are filtered out. As shown in Fig. 2, impact subgraphs in our production system typically contain fewer than 50 nodes, with the largest under 300, indicating that failures rarely cascade throughout the entire system. This makes persisting and analyzing these focused subgraphs more cost-efficient than processing whole-system telemetry data. The extracted subgraphs serve as inputs for incident detection and diagnosis, with the earliest subgraph of each issue persisted for ongoing model improvement.

C. Incident Detection

Incidents are detected from issues by analyzing their extracted impact subgraphs. During incident detection, GEM focuses on service entities within these subgraphs, as their states sufficiently reflect user experience impacts. When impact subgraphs contain non-service entities, these are omitted to create a subgraph containing only service entities and their relationships. In the following, we first describe the critical features engineered from these subgraphs for incident identification, then present our proposed model that flexibly incorporates these features to determine incident status. Finally, we explain how we evolve this model from scratch, addressing the challenge of potentially scarce incident records during initial training phases.

1) Critical Features of Incidents: We first introduce what information in impact subgraphs should be mined for incident detection. Specifically, whether an issue is an incident depends on its severity, which can be assessed mainly from two perspectives, namely the affected service attributes and the issue symptoms. Such information are critical features for incident detection. An example illustrating the mining process of these critical features is in Fig. 3. These features are selected after analyzing historical incident data and discussing with



(a) The topology of an impact subgraph. (b) An example of the node features presenting the affected service attribute information of the issue. (c) An example of the edge features presenting the issue symptom information, which is constructed using the latest telemetry data.

Fig. 3. An example of feature engineering to the impact subgraph of a fault injected to the simulation environment.

experienced on-call engineers, who confirmed these factors significantly influence their incident severity assessments.

Attributes of Affected Services. What are being affected needs to be figured out first for incident detection, which can be characterized by the attributes of affected service. Specifically, these service attributes are an important class of service meta information, which is usually organized and stored in the Configuration Management Database (CMDB) to manage services appropriately in a large-scale online system. The following service attributes are considered: (1) *Importance level*: This attribute helps indicate business criticality of a service. Critical services (e.g., payment processing) require immediate attention due to revenue or user experience impact. (2) *Provided functionality*: This attribute helps influence prioritization as customer-facing functionalities typically require faster resolution than internal administrative functions. (3) *Online status*: This attribute helps distinguish between online and offline services. (4) *Service role*: This attribute helps identify the infrastructure role of a service. Services of different roles (e.g., database or proxy) exhibit different incident patterns. (5) *Implemented fault tolerance strategy*: This attribute helps indicate resilience to failures, with less fault-tolerant services presenting higher incident risk. Detailed examples of these service attributes are in Appendix A.

Issue Symptoms. How seriously are the services being affected is also an important factor for incident detection, which can be characterized by issue symptoms. Issue symptoms can be described and calculated using the telemetry data of the abnormal entities and relationships. Generally, only faults that affect the latency or success rate of some key functions are considered as incidents, so we perform feature engineering on FC to describe the issue symptoms for incident detection. The features are selected through a systematic process combining domain expertise and historical incident analysis, guided by three principles: (1) capturing issue severity, (2) contextualizing anomalies relative to calling traffic patterns, and (3) identifying deviations relative to historical patterns. The features include: (1) *Binned FC values*: Discretized failure count values to capture non-linear relationships between FC and issue severity, with certain thresholds indicating critical situations. (2) *FC normalized by call count*: FC divided by call count per minute to obtain failure rates, as identical failure counts have different implications based on calling count volume. (3) *FC relative to historical baselines*: Ratio of current FC to its historical values to identify unusual spikes.

Putting Them Together in Impact Subgraphs. We represent affected services as nodes in the impact subgraph with node features $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times n}$, where n is the feature size. Relationship-level issue symptoms are represented as edge features $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times s}$, with s as the feature size. Additionally, global features $\mathbf{G} \in \mathbb{R}^m$ capture overall subgraph information, including subgraph size, important service count, and total FC count. For preprocessing, we apply one-hot encoding to categorical features, logarithmic transformation to handle long-tailed distributions, and Z-score normalization to prevent dimensional dominance.

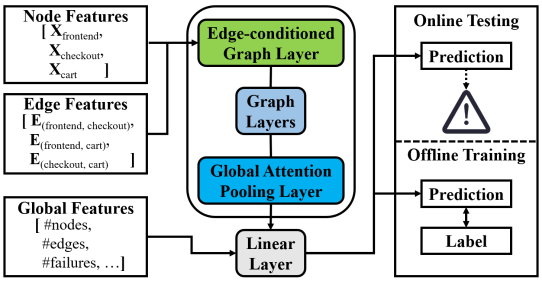


Fig. 4. An overview of the incident detection process by GEM with a graph based model to integrate diverse sources of features.

2) Detection Model Formulation: To accommodate varying impact subgraph sizes (as shown in Fig. 2) and complex node-edge interactions, we employ an edge-conditioned graph neural network for incident detection. As illustrated in Fig. 4, our model comprises: an edge-conditioned graph layer [34] that aggregates node and edge features, optional graph layers [35] for further processing hidden representations, a global attention pooling layer [36] that summarizes the entire impact subgraph, and a linear layer that produces output based on this summary and global features.

Suppose that the node features of nodes i, j are represented as $\mathbf{X}_i$, $\mathbf{X}_j \in \mathbb{R}^n$, respectively, and the edge features between them are represented as $\mathbf{E}_{(i,j)} \in \mathbb{R}^s$, the feed-forward process of edge-conditioned graph layer [34] to calculate node i's hidden representation $\mathbf{X}'_i \in \mathbb{R}^d$ with d representing the hidden dimension, is as follows:

$$\mathbf{X}'_i = \mathbf{X}_i \mathbf{\Theta}_1 + \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{X}_j (\text{reshape}(\mathbf{E}_{(i,j)} \mathbf{\Theta}_2)), \quad (4)$$

where $\mathcal{N}(i)$ denotes node i's neighborhood derived from $\mathcal{V}$ and $\mathcal{E}$, $\mathbf{\Theta}_1 \in \mathbb{R}^{n \times d}$ and $\mathbf{\Theta}_2 \in \mathbb{R}^{s \times (n \times d)}$ refer to trainable weights, and $\text{reshape}(\cdot)$ is a function that reshapes $\mathbf{E}_{(i,j)} \mathbf{\Theta}_2$ from a one-dimension vector with shape $\mathbb{R}^{(n \times d)}$ to a two-

dimension matrix with shape $\mathbb{R}^{n\times d}$ and thus the result can be further multiplied with X_j.

With the edge-conditioned graph layer, we obtain a summarized representation for each service considering both of its service attribute information, service-level issue symptom information and relationship-level issue symptom information. Afterwards, optional graph attention layers [35] are added to construct a deeper network to produce a more comprehensive representation for each service.

When the final hidden representation of each service is determined, the next step is to calculate an initial summarized representation $\mathbf{S}$ for the whole impact subgraph through a pooling operation on hidden representations of all the nodes. Specifically, suppose that the calculated representation $\mathbf{X}'_i$ represents the final hidden representation of node i, a global attention pooling layer [36] is formulated as:

$$\mathbf{S}' = \sum_{i\in\mathcal{V}} \mathrm{softmax}\left(\mathbf{X}'_i\mathbf{\Theta}_3\right)\odot\mathbf{X}'_i, \tag{5}$$

where $\mathbf{\Theta}_3 \in \mathbb{R}^{d\times 1}$ is a trainable weight to assign a score for each node, $\mathrm{softmax}(\cdot)$ is applied over node scores to decide which nodes are relevant to the task and $\odot$ denotes element-wise multiplication.

The calculated initial summarized representation $\mathbf{S}' \in \mathbb{R}^d$ is concatenated with the global features $\mathbf{G} \in \mathbb{R}^m$ to produce the signature $\mathbf{S} \in \mathbb{R}^{d+m}$ for the whole impact subgraph. Finally, $\mathbf{S}$ is fed to a linear layer with parameters $\mathbf{\Theta}_4 \in \mathbb{R}^{(d+m)\times 1}$, producing the incident score $\hat{y}$:

$$\mathbf{S} = \mathrm{concatenate}\left(\mathbf{S}', \mathbf{G}\right), \tag{6}$$

$$\hat{y} = \begin{cases} \mathrm{RAD}(\mathbf{S}) & \text{if } \mathbf{\Theta} \text{ has not been trained with enough data} \\ \mathbf{S}\mathbf{\Theta}_4 & \text{otherwise} \end{cases}. \tag{7}$$

The $\mathrm{RAD}(\cdot)$ denotes a retrieval augmented detection process for incident detection cold start. How to perform retrieval augmented detection and how to train $\mathbf{\Theta}$ will be elaborated in the following.

3) Detection Model Learning and Evolution: To train the incident detection model (i.e., $\mathbf{\Theta}$), we require training data with labels. Specifically, issues detected as incidents can be confirmed by operators, and if operators confirm that the result is correct, we can gain a positive sample, and otherwise a negative sample. By doing so, true positives and false positives of our model can be figured out. Then, during operators are writing an incident report, they need to fill in the root cause service and affected services, and then the corresponding impact subgraph is retrieved and attached to the report. By doing so, false negatives of our model can be figured out and labelled as positive samples. Other samples are labelled as negative samples (i.e., unimportant issues) by default.

Stage 1: Evolving Retrieval Augmented Detection for Cold Start. Despite the labelling process introduced above, we still cannot get sufficient labelled data to train the model at the beginning. The model will be prone to overfitting if trained on few data. To alleviate this problem, when few labelled data are collected, we use a retrieval augmented detection method (i.e., $\mathrm{RAD}(\cdot)$ in Equation 7) for the cold start of incident detection, and try to enable it to learn online from the gradually accumulated labelled data. Specifically, GEM continually keeps track of the signatures $\mathbf{S}$ calculated by the initialized model, and then gives a detection for a new observation based on its nearest neighbors [37] in the space of $\mathbf{S}$. This idea is inspired by the nearest neighbor based classification [38] for tabular data, which has been shown to be particularly useful for adaptation on evolving distributions [39], while here we need to perform online classification for graph data, so we calculate the nearest neighbors on the space of $\mathbf{S}$ derived by the GEM model. Specifically, for a given impact subgraph T and its signature $\mathbf{S}_T$ (calculated based on Equation 4∼6), the inference process of $\mathrm{RAD}(\cdot)$ in Equation 7 is formulated as:

$$\mathrm{RAD}(\mathbf{S}_T) = \frac{\mathrm{e}^{\sum_{(y_i=1)\wedge(\mathbf{S}_i\in N(\mathbf{S}_T))}\frac{1}{ED(\mathbf{S}_T,\mathbf{S}_i)}}}{\sum_{c\in\{0,1\}}\mathrm{e}^{\sum_{(y_j=c)\wedge(\mathbf{S}_j\in N(\mathbf{S}_T))}\frac{1}{ED(\mathbf{S}_T,\mathbf{S}_j)}}}. \tag{8}$$

Here, e is the Euler number, and $N(\mathbf{S}_T)$ denotes the several nearest neighbors of $\mathbf{S}_T$ in the space of $\mathbf{S}$, and $ED(\cdot)$ denotes the Euclidean distance between two representations. When $\hat{y}_T = \mathrm{RAD}(\mathbf{S}_T)$ is higher than 0.5 (i.e., over half of weighted nearest neighbors are incidents), GEM considers the input T implies an incident.

To agilely make full use of the newly collected data, the space of $\mathbf{S}$ is incrementally expanded as the model is being deployed for incident detection. Therefore, this retrieval augmented detection based incident detection process is highly adaptable to new data, as it can immediately incorporate newly labeled samples into the incident detection process, and thus it well suits the cold start stage of incident detection when training data are few. However, since $\mathbf{\Theta}$ is not properly trained on a sufficient amount of incident data, the representation ability of the issue signature $\mathbf{S}$ is still weak, hindering its effectiveness for incident detection. This problem will be mitigated in the following stage when data have been collected for a period of time.

Stage 2: Periodical Model Re-training for Continual Learning. After the model has been deployed for a period of time, we can get more labeled data to train a more accurate model. The training is performed periodically (e.g., every two weeks) to capture the characteristics of new data.

During the periodical model re-training phase, we adopt the cross-entropy as the loss function and Adam [40] as the optimizer to train all the model parameters $\mathbf{\Theta}$. With y representing the label for the impact subgraph and $\sigma(\cdot)$ representing a sigmoid operation, the training loss function is defined as:

$$L(y,\hat{y}) = -y\log\left(\sigma(\hat{y})\right) - (1-y)\log\left(1-\sigma(\hat{y})\right). \tag{9}$$

During the online inference phase, samples with an incident score $\hat{y}$ higher than τ will be detected as an incident and corresponding alerts will be sent to on-call operators. In practice, only a minority of impact subgraphs might imply an incident, so the model will tend to give a low $\hat{y}$ and then

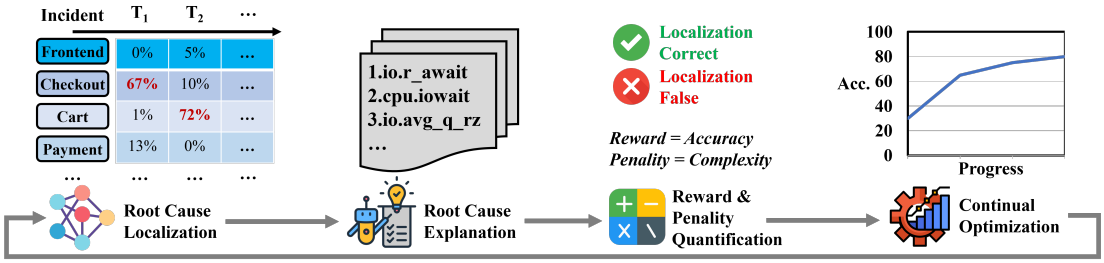


Fig. 5. An overview of the incident diagnosis process by GEM with explanation driven online continual optimization.

detect every sample as a negative sample with the training objective in Equation 9. To better handle this data imbalance problem, we select a threshold τ which can make detection results maximize the F1-score in training data. F1-score is the harmonic mean of Precision and Recall, weighting them equally. It describes the effectiveness of a detection model, and can well consider the data imbalance problem.

D. Incident Diagnosis

After an incident is detected, we need to further diagnose its root cause. Fig. 5 presents an overview of the incident diagnosis process powered by GEM.

1) Diagnosis Model Formulation: We introduce how GEM formulates the incident diagnosis model, covering both root cause localization and explanation.

Root Cause Localization. When diagnosing an incident, operators typically investigate along entity relationships to find diagnosis clues leading to the root cause entity. GEM models this process using the PageRank algorithm [41], where the PageRank vector $\vec{v}$ represents root cause scores of different entities, and the transition probability matrix $\mathbf{P}$ is calculated using diagnosis clues defined on telemetry data. Specifically, a **diagnosis clue** is a function that takes a direction (src, des) as input and outputs a tendency $p \in \mathbb{R}^+$ based on telemetry k. This tendency correlates with the possibility that the destination entity des is close to the root cause.

Table I illustrates the expressions of diagnosis clues in GEM. We use $D_{i,k}$ to denote the day-by-day difference of normalized $\mathbf{X}_{i,k}$, and $D_{(i,j),k}$ for the day-by-day difference of normalized $\mathbf{E}_{(i,j),k}$. These clues direct attention to entities with more severe symptoms, while the PageRank integration can emphasize entities affecting multiple abnormal entities. To address symptom cascading across entities, GEM eliminates interference from this cascading effect when calculating the staying clue $\text{clue}_{k,2}$ by subtracting the effect of dependent entities from symptom strength. Additionally, GEM implements backward clue $\text{clue}_{k,3}$ to reduce visits to less abnormal entities while prioritizing abnormal siblings. Parameter β is used to to control the backward visit frequency, as the sum operation tends to inflate backward edge clue values.

Then, suppose that $\mathcal{K}$ is the set of all diagnosis telemetry considered useful, and the c^{th} clue ($c \in \mathcal{C}$ and $\mathcal{C} = [1, 6]$) on the telemetry $k \in \mathcal{K}$ is defined as $\text{clue}_{k,c}$ and its expression for the direction from entity i to entity j is represented as $\text{clue}_{k,c}(i, j)$, the transition probability matrix $\mathbf{P}$ is calculated through finding the strongest clues as follows:

$$\mathbf{M}_{i,j}=\max\Big(\{\alpha_k \text{clue}_{k,c}(i,j) | k \in \mathcal{K}, c \in \mathcal{C}, (i,j) \in \text{dom}(\text{clue}_{k,c})\}\Big) \tag{10}$$

$$\mathbf{P}_{i,j} = \frac{\mathbf{M}_{i,j}}{\sum_j \mathbf{M}_{i,j}}. \tag{11}$$

Here, $\max(\cdot)$ is a maximization operation applied over the set containing different clues (if the set is an empty set, the output is 0), α_k stands for the strength parameter for $clue_{k,c}$, and $\text{dom}(\text{clue}_{k,c})$ represents the domain of $\text{clue}_{k,c}$. Specifically, for each possible direction (i, j), GEM iterates over all $k \in \mathcal{K}$ and $c \in [1, 6]$, and if (i, j) falls in the domain of a specific $\text{clue}_{k,c}$, the corresponding $\text{clue}_{k,c}(i, j)$ will be accumulated to a set, and then GEM will find the strongest clue in the set for transition probability calculation.

With the transition probability matrix $\mathbf{P}$ defined above, the PageRank equation can be formulated as:

$$\vec{v} = (1 - \lambda)\mathbf{P}\vec{v} + \lambda\vec{u}, \tag{12}$$

where λ denotes the damping parameter and $\vec{u}$ denotes the additional teleportation vector. By default, $\vec{u} = [\frac{1}{|\mathcal{V}|}, \frac{1}{|\mathcal{V}|}, \cdots, \frac{1}{|\mathcal{V}|}]^T$.

The root cause localization result $\hat{r}$ is then calculated as:

$$\hat{r} = \arg\max_i \vec{v}_i. \tag{13}$$

Root Cause Explanation. After the root cause is located, GEM also provides an explanation to help operators better understand it. Specifically, an explanation is defined as a subset of telemetry which can well describe what is happening in a specified entity. Our key idea is that, if the symptom strength $D_{i,k}$ of a telemetry (e.g., telemetry k) of an entity (e.g., entity i) is significantly distinguished from the the symptom strengths of the same telemetry of other entities in the same impact subgraph, this telemetry k can well represent what is happening in the entity i. Based on this idea, the explanation power of a telemetry k for an entity i in an impact subgraph T is defined as,

$$\delta(T, i, k) = D_{i,k} - max_{j \in \mathcal{V}_T} D_{j,k}, \tag{14}$$

where $\mathcal{V}_T$ denotes the node set of T. Then, the explanation for a specified entity i is defined as the set $\{k | \delta(T, i, k) > 0\}$. If we specify the localization result as the entity to be explained, then the derived explanation can explain why it is likely to be the root cause. Moreover, if it turns out that the localization

TABLE I
THE EXPRESSIONS AND DESCRIPTIONS OF DIAGNOSIS CLUES USED FOR INCIDENT DIAGNOSIS.

Clue	Domain	Expression	Description
Edge Clues (clues calculated using relationship-level telemetry data):			
$\text{clue}_{k,1}(i,j)$	$\{(i,j)\|(i,j)\in\mathcal{E}\}$	$D_{(i,j),k}$	Forward clues, which favor a destination with serious symptoms.
$\text{clue}_{k,2}(i,i)$	$\{(i,i)\|i\in\mathcal{V}\}$	$\max_z D_{(z,i),k} - \max_z D_{(i,z),k}$	Staying clues for stopping at a node with serious symptoms.
$\text{clue}_{k,3}(j,i)$	$\{(i,j)\|(i,j)\in\mathcal{E}\}$	$\beta(\sum_z D_{(i,z),k} - D_{(i,j),k})$	Backward clues, which revisit a source with serious symptoms.
Node Clues (clues calculated using entity-level telemetry data):			
$\text{clue}_{k,4}(i,j)$	$\{(i,j)\|(i,j)\in\mathcal{E}\}$	$D_{j,k}$	Forward clues, which favor a destination with serious symptoms.
$\text{clue}_{k,5}(i,i)$	$\{(i,i)\|i\in\mathcal{V}\}$	$D_{i,k}$	Staying clues for stopping at a node with serious symptoms.
$\text{clue}_{k,6}(j,i)$	$\{(i,j)\|(i,j)\in\mathcal{E}\}$	$D_{i,k}$	Backward clues, which revisit a source with serious symptoms.

result by GEM is wrong and the actual root cause is another entity, we can still use this method to calculate explanation for the actual root cause, and then optimize GEM automatically based on the explanation, which will be detailed below.

2) Diagnosis Model Learning and Evolution: The keys of our GEM incident diagnosis model are the set of diagnosis telemetry $\mathcal{K}$ and their associated strength parameters $\{\alpha_k|k\in\mathcal{K}\}$ for the calculation of Equation 10. This section explains how GEM learns and evolves them gradually.

Cold Start. $\mathcal{K}$ is initialized as a set containing only FC (i.e., failure count per minute as introduced in Section III-B). If FC is not monitored or provided, GEM uses the alert count per minute (i.e., the number of detected anomalies in all telemetry data from a specific entity or relationship) as a substitution of FC. Then α_{FC} is initialized as 1. This cold start setting enables GEM to be usable for systems with arbitrary telemetry data.

Reward & Penalty Quantification. To quantify the quality of $\mathcal{K}$ and $\mathcal{A}=\{\alpha_k|k\in\mathcal{K}\}$ for incident diagnosis, GEM continually retrieves the root cause service results included in the incident reports after a new incident report is finished, and then calculates the reward and penalty based on the root cause service results and the localization results by GEM. Specifically, the reward of $\mathcal{K}$ and $\mathcal{A}$ for incident diagnosis is calculated as their localization accuracy for historical incidents, which is formulated as:

$$Reward(\mathcal{K},\mathcal{A},\mathcal{T},\mathcal{R}) = \frac{|\{T|\hat{r}_T^{\mathcal{K},\mathcal{A}}=\mathcal{R}_T, T\in\mathcal{T}\}|}{|\mathcal{T}|}, \quad (15)$$

where $\mathcal{T}$ and $\mathcal{R}$ are all historical impact subgraphs and their corresponding root cause entities, $|\cdot|$ denotes the cardinality of a set, $\hat{r}_T^{\mathcal{K},\mathcal{A}}$ denotes the root cause localization result using telemetry set $\mathcal{K}$ and parameters $\mathcal{A}$ for an impact subgraph T, and $\mathcal{R}_T$ is its ground-truth root cause based on reports.

Beyond assigning a high reward to a well-behaved model, GEM also penalizes a too complex model to avoid over-fitting. Specifically, the penalty is achieved via an L1-normalization on the $\mathcal{A}$. L1-normalization promotes sparsity in the solution [42], which aligns with our goal of identifying the most significant telemetry while naturally eliminating less important ones. Specifically, the penalty is calculated as:

$$Penalty(\mathcal{A}) = \max(\sum_{\alpha_k\in\mathcal{A}}|\alpha_k|, 1). \quad (16)$$

Here, the max operation is used to clip the penalty calculation since endlessly minimizing the model complexity is meaningless. Therefore, only when the diagnosis model is more complex than the initial diagnosis model for cold start (whose penalty is 1 since $|\alpha_{\text{FC}}|=1$ at the beginning), GEM will penalize the diagnosis model.

Continual Optimization. $\mathcal{K}$ and $\mathcal{A}$ are continually optimized after GEM has been deployed for incident diagnosis. Specifically, after GEM has calculated a root cause localization result for a specific incident, the correctness of the result can be confirmed from its follow-up incident report. Then, if the result is incorrect, a continual optimization of the GEM diagnosis model will be triggered. During the optimization, GEM will first optimize $\mathcal{K}$, and then $\mathcal{A}$ accordingly.

The optimization of $\mathcal{K}$ is aided by the root cause explanation process introduced in Section III-D1. Specifically, when operators complete incident reports, they provide ground truth information about root causes. Then, for an impact subgraph T whose root cause r has been mis-located by GEM (as identified in the incident reports), GEM searches for the telemetry k with the highest explanation power $\delta(T,r,k)$ calculated using Equation 14, and then adds it to $\mathcal{K}$. This human-in-the-loop optimization process is formulated as:

$$\mathcal{K}_t = \mathcal{K}_{t-1}\cup\{\arg\max_k \delta(T,r,k)\}, \quad (17)$$

where the original set of diagnosis telemetry at time step $t-1$ is $\mathcal{K}_{t-1}$, and the optimized set of diagnosis telemetry at time step t is $\mathcal{K}_t$.

Then, the optimization of $\mathcal{A}$ is driven by the reward and penalty quantification introduced in Equation 15 and 16, which is formulated as:

$$\mathcal{A}_t = \arg\max_{\mathcal{A}}\Big(Reward(\mathcal{K}_t,\mathcal{A},\mathcal{T}_{0:t},\mathcal{R}_{0:t}), -Penalty(\mathcal{A})\Big), \quad (18)$$

where $\mathcal{A}_t=\{\alpha_k|k\in\mathcal{K}_t\}$ is the set of strength parameters to be optimized for $\mathcal{K}_t$ obtained in Equation 17, and $\mathcal{T}_{0:t}$ and $\mathcal{R}_{0:t}$ are reported incidents and their root causes up to time step t. This is a multi-objective optimization problem, where the first objective is to maximize the reward and the second objective is to minimize the penalty. GEM applies Multi-objective Tree-structured Parzen Estimator (MOTPE) [43] to optimize these two objectives simultaneously. We choose MOTPE here because it offers an efficient approach to handle multi-objective optimization problems. Given a number of sampling trials, MOTPE can approximate the Pareto front of $\mathcal{A}$. Specifically, MOTPE constructs two probability density functions, i.e., $l(\mathcal{A})$ and $g(\mathcal{A})$. $l(\mathcal{A})$ models the density of good observations (i.e., the samples of $\mathcal{A}$ that achieve a reward higher than a quantile-based threshold or a penalty lower than a quantile-based threshold), and another $g(\mathcal{A})$ models the density of the remaining observations. MOTPE then samples candidate

solutions from $l(\mathcal{A})$ and selects the best candidate based on the Expected Hypervolume Improvement criterion [43], which aims to maximize the hypervolume contribution of new trials. This iterative process helps approximate the Pareto front effectively. After a given number of trials, GEM selects $\mathcal{A}_t$ as the trial with the highest reward from the Pareto front.

To improve the optimization of $\mathcal{A}$ within an limited number of sampling trials, GEM also maintains a memory of high-quality historical trials, including $\mathcal{A}_{t-1}$ and $\mathcal{A}_0$. Then, GEM expands them by setting newly-added strength parameters as 0, and warms up the MOTPE optimization with these trials.

With the continual optimization triggered when the root causes are mis-predicted, the incident diagnosis model can gradually and automatically consider more useful telemetry for incident diagnosis, adapting to new failures and systems effectively. This model performance-triggered approach allows the model evolves specifically in response to diagnostic errors rather than following an arbitrary retraining schedule.

IV. EVALUATION

In this section, we carry out experiments to evaluate our proposed framework. We aim at answering the following research questions:

- **RQ1. Incident Detection Performance:** How effective is GEM for incident detection compared with state-of-the-art incident detection methods?
- **RQ2. Incident Detection Ablation Study:** How useful are different sources of information extracted by GEM for incident detection?
- **RQ3. Incident Diagnosis Performance:** How effective is GEM for incident diagnosis compared with state-of-the-art incident diagnosis methods?
- **RQ4. Incident Diagnosis Ablation Study:** How useful are different designs of GEM for incident diagnosis?
- **RQ5. Efficiency:** How efficient is GEM?

A. Experiment Setting

1) Datasets: We evaluate GEM using both production environment and open-sourced datasets. The production datasets contain issues extracted from the WeChat online system. Two experts (the third and fourth authors of this paper) in the Wechat operator team built a data labelling system for triggered incident alerts, collecting labels from over 50 Wechat engineers to construct the datasets. The incident detection evaluation datasets (RQ1∼2) were collected from 2020-6-15 to 2020-9-28 and divided chronologically into parts A, B, and C. Each dataset contains **over 3K** system issues. The earliest training cases in dataset A are used to evaluate the cold start performance, denoted as "Cold Starting". Among all incidents, **77** contain detailed diagnosis information in their incident reports. Incident reports are unstructured text-based descriptions written by operators, typically including a clear timeline, detailed root cause diagnosis, thorough impact scope analysis, and subsequent optimization plans. These 77 incidents are used for incident diagnosis evaluation (RQ3∼4). We also evaluated GEM on open-sourced datasets from prior studies [10], [15], including datasets from Online Boutique (OB) [44] and the AIOps Challenge 2021 [45]. These open-sourced datasets are used only for fault diagnosis evaluation (RQ3∼4), as RQ1∼2 focuses on incident detection rather than fault detection. To address transposability concerns, we provide detailed characterization of our production datasets in Appendix A, including service attributes, telemetry data, and visualized examples. With this information, researchers can prepare similar data structures in their environments to apply our method. Our artifact also includes synthetic data (generated by injecting faults into the Online Boutique benchmark system) that demonstrates the usage of GEM step by step, helping readers understand our methodology and facilitating adaptation to other systems, even though these synthetic datasets are not suitable for comprehensive model comparison on incident detection. More details are in Appendix A.

2) Evaluation Metrics: For the evaluation of incident detection (RQ1∼2), we use **Precision, Recall, and F1-score**. Precision measures the percentage of incidents that are correctly detected over all positives triggered by a method, while Recall measures the percentage of incidents that are correctly detected over all actual incidents, and F1-score is the harmonic mean of Precision and Recall. That is, F1-score $= \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$. We also draw the **Precision-Recall (PR) curve** for RQ2, and use the area under it (also known as **PR-AUC**) to evaluate the usefulness of different sources of information. The PR curve can give an informative picture of the performance of the incident detection results, and it is well suited for imbalanced data [46] (i.e., incidents only accounts for a few cases).

For the evaluation of incident diagnosis (RQ3∼4), we use Top-1 Accuracy (**A@1**), which denotes the probability that the root cause is within the top-1 of the ranked root cause list based on the calculated root cause scores. Prior methods generally require training data to first train the incident diagnosis model offline. Therefore, they need to split the data, and then use the former data as the training data and the left data as the test data. Therefore, we denote the A@1 on test data as **Test A@1**. However, Test A@1 assumes training data are always ready, and thus neglects the preparation cost of training the model. Therefore, we also calculate **Life A@1**, which stands for the A@1 throughout the model evolution lifecycle. Specifically, we calculate A@1 on all cases to gain the Life A@1. It is noted that for offline training based methods that do not consider cold start, the A@1 for the train cases is considered as 0 since their models are not ready when the training data are still being collected.

3) Hyper-Parameter Setting: The hyper-parameters are set with a cross-validation approach, and the detailed setting of each hyper-parameter is illustrated in the Appendix C. Besides, we provide a parameter sensitivity study in Appendix C.

B. RQ1: Incident Detection Performance

Experimental Plan. To validate the performance of GEM in incident detection, we compare it with state-of-the-art methods that detect incidents using telemetry data, including AirAlert [2] and Warden [3]. For each issue, we construct an input for these baselines by setting the values of non-alerting signals outside an impact subgraph as 0. Some additional methods are also considered for comparison, such as

TABLE II
THE COMPARISON OF GEM WITH STATE-OF-THE-ART INCIDENT DETECTION METHODS.

Datasets	Cold Starting			A			B			C		
Methods	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
AirAlert	-	-	-	0.56	0.24	0.34	0.38	0.24	0.30	0.38	0.16	0.23
Warden	-	-	-	0.36	0.21	0.26	0.26	0.2	0.23	0.21	0.10	0.13
GEM	**0.71**	**0.81**	**0.75**	**0.85**	**0.86**	**0.85**	**0.87**	**0.87**	**0.87**	**0.81**	**0.88**	**0.85**

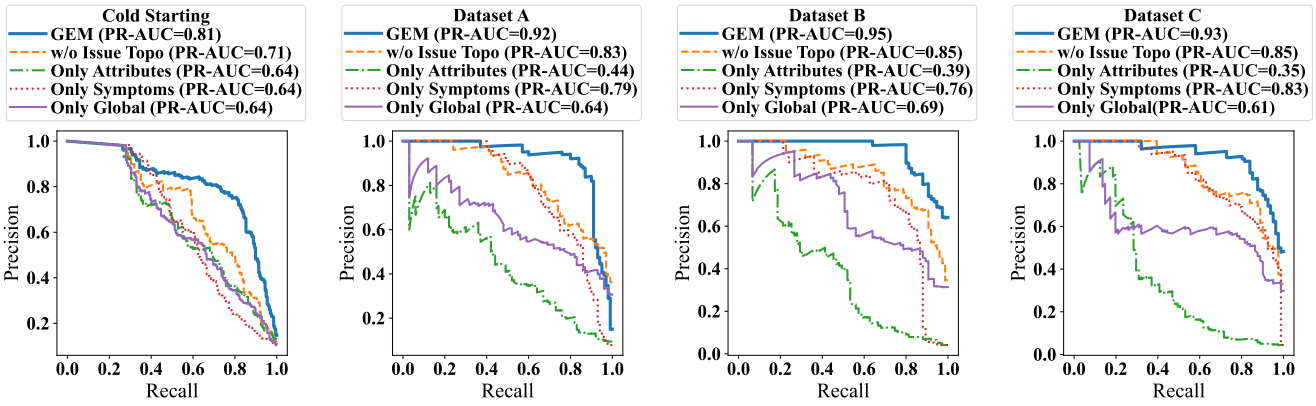


Fig. 6. The PR curves for the efficacy of different information and features at different evolution stages and datasets.

TopoMAD [6], ART [15], Eadro [9], and Hades [5], but we find them unsuitable due to the following reasons. (1) **Scope:** These methods target at fault detection rather than incident detection. (2) **Requirement:** They require continuous whole-system modeling using telemetry data from all entities, which is impractical in WeChat system where outdated telemetry data are routinely deleted to manage operational costs.

Quantitative Results. Table II presents the incident detection performance across different methods. GEM consistently outperforms baseline methods, achieving a relative improvement of 150%∼270% in F1-score over the best baselines across different datasets. Additionally, GEM provides a viable cold start option for incident detection when training data are still being collected. As shown in the "Cold Starting" results in Table II, GEM achieves acceptable incident detection performance (i.e., an F1-score of 0.75) even when learning from limited but gradually expanding training data, while prior methods lack this capability and must wait until sufficient training data are available.

Discussion of Results. The superior performance of GEM can be attributed to the expressiveness of the extracted impact subgraphs. Large-scale online systems contain enormous entities, creating a vast feature space (entity-telemetry tuples) that makes incident detection learning extremely challenging. Previous methods [2], [3] typically address this by pre-selecting incident-indicating features, assuming certain features work well for most incidents. However, since incidents can occur anywhere in the system, incident-indicating features for specific incidents may not be universally applicable. If such features are filtered out during feature selection, detection performance suffers significantly. In contrast, the impact extraction phase of GEM performs targeted feature selection based on extracted impact subgraphs, effectively reducing the feature space to only include entities closely related to ongoing issues. This graph-based issue representation approach significantly enhances incident detection capabilities in large-scale online systems.

C. RQ2: Incident Detection Ablation Study

Experimental Plan. In this section, we conduct an ablation study to validate the contribution of different information sources to incident detection performance. We examine four key components: issue topology information (encoded in $\mathcal{E}$), affected service attribute information (encoded in $\mathbf{X}$), issue symptom information (encoded in $\mathbf{E}$), and issue global feature information (encoded in $\mathbf{G}$). To isolate the effect of each component, we create several variants of GEM: (1) **w/o Issue Topo**: Removes the issue topology $\mathcal{E}$ and instead uses a summation aggregator to combine affected service attribute features ($\sum_{i\in\mathcal{V}}\mathbf{X}_i$) and issue symptom features ($\sum_{(i,j)\in\mathcal{E}}\mathbf{E}_{i,j}$). These aggregated representations are normalized and concatenated with global features $\mathbf{G}$ to form $\mathbf{S}$, which is then used in the incident detection model (Equation 7). (2) **Only Attributes**: Uses only service attribute information $\mathbf{X}$. (3) **Only Symptoms**: Uses only issue symptom information $\mathbf{E}$. (4) **Only Global**: Uses only issue global feature information $\mathbf{G}$.

Quantitative Results. Fig. 6 presents the incident detection performance across all variants. The results demonstrate that removing the issue topology information ("w/o Issue Topo") leads to performance degradation. Further performance drops are observed when using individual information sources in isolation ("Only Attributes", "Only Symptoms", and "Only Global"). This pattern holds consistent for both the cold start online learning stage and the continual learning stage. Notably, using only affected service attributes ("Only Attributes") results in particularly poor incident detection performance. In datasets A and C, the "w/o Issue Topo" variant shows only marginal improvement over the "Only Symptoms" variant.

Discussion of Results. The experimental results highlight several important insights. First, all information sources contribute significantly to incident detection performance, with

TABLE III
THE COMPARISON OF GEM WITH STATE-OF-THE-ART INCIDENT DIAGNOSIS METHODS.

Method	Production	OB				AIOps 2021			
	No Split	train:test=3:7		train:test=6:4		train:test=3:7		train:test=6:4	
	Life A@1	Life A@1	Test A@1	Life A@1	Test A@1	Life A@1	Test A@1	Life A@1	Test A@1
MicroHECL	0.545	0.119	0.109	0.119	0.107	0.429	0.457	0.429	0.426
DiagFusion	-*	0.233	0.333	0.124	0.310	0.279	0.398	0.233	0.582
Dejavu	-*	0.331	0.473	0.164	0.411	0.408	0.583	0.161	0.402
Eadro	-*	0.217	0.310	0.055	0.137	0.150	0.214	0.063	0.157
DeepHunt	-*	0.562	0.803	0.333	0.833	0.550	0.785	0.304	0.759
ART	-*	0.467	0.667	0.267	0.667	0.536	0.766	0.289	0.722
GEM	**0.987**	**0.829**	**0.857**	**0.829**	**0.857**	**0.789**	**0.861**	**0.789**	**0.796**

* These baselines are unavailable for our large-scale online system since they target at small-scale online systems. Specifically, they need all the telemetry data (including normal and abnormal telemetry data) from the whole system to perform continuous whole-system modelling, but we cannot continually persist such an overwhelming volume of data since outdated telemetry data are deleted routinely to save operation cost in our large-scale WeChat system.

their combined use yielding superior results compared to any subset. Second, the issue topology plays a crucial role in organizing and contextualizing the other information sources. Without proper structural organization provided by the issue topology, the utility of service attributes, symptoms, and global features is substantially diminished. The particularly poor performance of the "Only Attributes" variant demonstrates that affected service attributes provide limited value when disconnected from their corresponding issue symptoms and topological context. The minimal improvement observed when using all information without topology (compared to symptoms alone) further emphasizes that proper integration of information sources is as important as the information itself. In conclusion, all information sources extracted by GEM are essential for effective incident detection across both online learning for cold start and periodical re-training for continual learning, with the issue topology serving as a critical framework for integrating these diverse information sources.

D. RQ3: Incident Diagnosis Performance

Experimental Plan. To validate the performance of GEM in incident diagnosis, we compare it with state-of-the-art incident diagnosis methods, including MicroHECL [18], DiagFusion [7], Dejavu [8], Eadro [9], DeepHunt [10], and ART [15]. We assess the methods using Life A@1 and Test A@1 across different datasets and train-test splits (details of these metrics are in Section IV-A2).

Quantitative Results. Table III presents the incident diagnosis performance across all methods. GEM consistently achieves superior localization accuracy compared to prior methods, with relative improvements of 43.45%∼159.54% in Life A@1 and 2.88%∼9.67% in Test A@1 over the best baselines across different datasets and train-test splits. The performance gap is particularly pronounced in Life A@1 metrics, demonstrating the strong cold-start capability of GEM without requiring operators to prepare extensive training datasets for offline warm-up.

Discussion of Results. The superior performance of GEM can be attributed to two key factors. First, by extracting impact subgraphs and treating them as first-class citizens for analysis, GEM significantly narrows the scope for root cause searching, improving incident diagnosis accuracy. Second, the root cause searching heuristic defined in Equation 10 effectively simulates operator practices in root cause localization while continuously learning from past cases. A notable advantage of GEM over existing root cause localization algorithms [7]–[10], [15] is its practicality for real-world large-scale online systems. While many existing approaches rely on whole-system modeling and continuous data persistence, which limits their applicability in large-scale environments, GEM avoids these limitations by focusing on impact subgraphs, enabling more cost-efficient data persistence. This approach is particularly valuable for large-scale online systems that support numerous users, where incidents are costly and require rapid resolution. In summary, GEM demonstrates high effectiveness in incident diagnosis for online systems through its outstanding root cause localization accuracy, scalability for large-scale environments, and robust cold-start capability.

E. RQ4: Incident Diagnosis Ablation Study

Experimental Plan. We further perform an ablation study to validate the effectiveness of different design choices of GEM for incident diagnosis. Specifically, we compare the full version of GEM with its following variants: (1) **w/o Impact Extraction**: We remove the impact extraction process of GEM and directly use the whole-system topology and telemetry data to locate root cause. (2) **w/o Continual Optimization**: We remove the continual optimization process (i.e., Equation 17 and 18) of GEM, and continually use the initialized version of GEM to perform root cause localization. (3) **w/o $\mathcal{K}$ Selection**: We remove the $\mathcal{K}$ selection process (i.e., Equation 17), and directly set $\mathcal{K}$ as all available telemetry for root cause localization. (4) **w/o Penalty Minimization**: We remove the penalty quantification process (i.e., Equation 16) and its minimization during the continual optimization (i.e., Equation 18). Moreover, to quantify the impact of penalty minimization on preventing over-fitting, we measure the test-train difference, i.e., the gap between the final life A@1 (test performance) and its achieved Reward (training performance from Equation (15)). (5) **w/o Warm Up**: We remove the memory of high-quality historical trials, and do not perform warm up during continual optimization. (6) **w/ L2 Penalty**: We replace the L1-normalization into L2-normalization.

Quantitative Results. Table IV presents the evaluation results of GEM and its variants on incident diagnosis performance. We can see that the full version of GEM outperforms all of its variants on OB and AIOps 2021. For the Production dataset, GEM performs comparably to its variants, with the

TABLE IV
THE LIFE A@1 OF DIFFERENT VARIANTS OF THE GEM INCIDENT DIAGNOSIS MODEL.

Variant	Production	OB	AIOps 2021
w/o Impact Extraction	-*	0.505	0.579
w/o Continual Optimization	0.987	0.762	0.774
w/o $\mathcal{K}$ Selection	0.987	0.733	0.774
w/o Penalty Minimization	0.987	0.733	0.759
w/o Warm Up	0.987	0.738	0.774
w/ L2 Penalty	0.987	0.810	0.789
Full GEM	**0.987**	**0.829**	**0.789**

* We cannot retrieve the whole-system telemetry data and topology for this variant on production data since outdated telemetry data are deleted to save operation cost. Only impact subgraphs are persistent for analysis.

cold-start version already working effectively on the WeChat online system.

Discussion of Results. With impact extraction, the root cause searching scope can be more focused, so the root cause localization accuracy of GEM is improved. With continual optimization (including $\mathcal{K}$ selection and strength parameter optimization), the feedback can guide the GEM incident diagnosis model to select useful telemetry and tune the corresponding strength parameters, so the root cause localization accuracy of GEM is improved. With penalty minimization, the complexity of the GEM incident diagnosis model is effectively controlled to avoid over-fitting. Specifically, GEM shows an average test-train difference of 0.029, compared to 0.052 when penalty minimization is disabled. This smaller gap confirms that penalty minimization reduces over-fitting in the incident diagnosis model. Besides, the choice of normalization (i.e., L1 or L2) does not substantially impact the final performance of the GEM model, but generally, L1-normalization is slightly better. With warm up, the optimization can re-use the knowledge of historical optimization trials, so the root cause localization accuracy of GEM is improved.

As for the Production dataset, GEM is on par with its variants. Specifically, the cold-start version of GEM has already worked very well on the WeChat online system. This is because the WeChat software engineer team has done a great job in improving the system design and implementation to make failures obvious and thus easier to diagnose. Therefore, for a well-engineered system where incidents will usually manifest as failures, it is okay to directly use the cold-start version of GEM for incident diagnosis. However, adding continual optimization can bring additional benefits when systems are not well engineered, so it is still recommended to use the full version of GEM for incident diagnosis.

F. RQ5: Efficiency

Experimental Plan. To validate the efficiency of GEM, we measure the execution time of its key steps, including impact extraction, incident detection, and incident diagnosis. We use the datasets introduced in Section IV-A1 (and detailed in Appendix A) to conduct these measurements on a server with 32 Intel(R) Xeon(R) E5-2667 v4 CPUs and 251GB RAM. To better understand the efficiency of GEM, we also measure the execution time of some baselines, and display the results together with that of GEM. Specifically, we use AirAlert and Warden as the efficiency baselines for incident detection, and DeepHunt and ART as the efficiency baselines for incident diagnosis. The purpose of this section is to demonstrate that the efficiency of GEM meets practical deployment requirements.

TABLE V
THE EXECUTION TIME OF DIFFERENT PROCEDURES OF GEM.

Procedure	Stage 1 Per-Sample Detection	Stage 2 Offline Training	Stage 2 Per-Sample Detection	Per-Sample Diagnosis*	Diagnosis Optimization*
AirAlert	-	0.73min	0.09ms	-	-
Warden	-	0.85min	0.07ms	-	-
DeepHunt	-	-	-	142.17ms	1.03min**
ART	-	-	-	478.01ms	15.26min**
GEM	3.63ms	1.17min	0.29ms	9.47ms	1.80min

* These results are measured and averaged across OB and AIOps 2021 for a direct and fair comparison between GEM and DeepHunt/ART. On the production data, the per-sample diagnosis time of GEM is 6.44ms, which is also within the millisecond scale, and thus meets the practical requirement.
* Unlike GEM, which employs online learning, DeepHunt and ART require offline learning on a training dataset. The optimization time of DeepHunt and ART is measured with a train-test split of 3:7.

Quantitative Results. For impact extraction by GEM, the completion time is generally within one second on WeChat production data, which varies based on the number of detected anomalies (detailed values under different stress conditions are provided in Appendix E). For incident detection and diagnosis, as shown in Table V, the online detection time by GEM is comparable to the incident detection baselines, and the online diagnosis time by GEM is superior to the incident diagnosis baselines. Since the efficiency of GEM remains within the millisecond-scale requirement for real-time processing, it is practical for real-world deployment. The offline incident detection training requires less than 2 minutes on average across datasets (ranging from 3222 to 5168 samples), while the diagnosis optimization process also completes within 2 minutes on average across datasets (ranging from 133 to 210 samples). These efficiency results are either comparable to or superior to prior approaches.

Discussion of Results. The measured performance of GEM satisfies practical deployment constraints. Specifically, the per-sample processing time witinin the millisecond scale enables real-time incident response, while the training and optimization time under 2 minutes constitutes little overhead given their infrequent execution (biweekly retraining for incident detection and on-demand optimization for incident diagnosis). Importantly, since these computational costs are relatively small in absolute terms, the efficiency of GEM will remain practical even as the datasets grow larger during system evolution. Overall, the results demonstrate the suitability of GEM for production deployment in large-scale systems.

V. DISCUSSIONS

A. Success Stories

GEM has been deployed in production at our organization since July 2020. In the initial three months, GEM ran in parallel with the existing incident management mechanism, which is based on manually-configured rules. This parallel deployment strategy allowed us to directly compare the performance of GEM against the original incident management

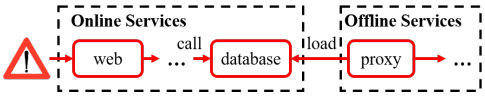


Fig. 7. A simplified impact subgraph for Real Case I.

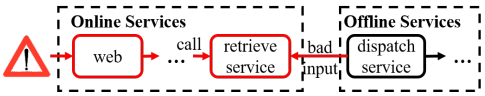


Fig. 8. A simplified impact subgraph for Real Case II.

mechanism. During a comprehensive three-month evaluation period, we collected data from incidents affecting our production environment. The results show that GEM has greatly helped operators in mitigating anomaly storms and accelerating the incident management. Specifically, at the incident detection phase, about **95%** of insignificant issues are filtered out. Moreover, at the incident diagnosis stage, the search space of root causes is narrowed down by **96%**, since the average size of an incident impact subgraph in our system is 26 while only one of them is recognized as the root cause. With this help, operators can focus on the key information of incidents and diagnose an incident more quickly.

Besides, according to incident reports in September 2020 (this month is the last month that GEM is deployed together with the original rule-based incident management mechanism in WeChat), in **68%** of the cases, GEM detected the incidents earlier than the manually configured alerts, and in **88%** of the recorded cases, GEM detected the incidents earlier than the time that an operator reported the corresponding incident[1]. In general, these results show that GEM can greatly help accelerate the detection and reporting of incidents in large-scale online systems.

We also routinely communicate with our operator team to understand the impact of GEM on their daily operation efficiency. Feedback from our operator team has been particularly positive. As noted by the leader of our operator team: "Before this approach, our team was constantly overwhelmed by alert storms that made it difficult to identify critical issues. Now, we can focus on what truly matters." Another operator pointed out that, "What impresses me most about this system is its ability to detect incidents before my configured alerting rules. In one particularly critical case, it identified a potential overload issue nearly 15 minutes before our configured alerts."

Case Studies. The following are some real use cases of GEM. The confidential details are anonymized.

Real Case I: This is an incident caused by an overload to a database service. At the beginning of this incident, an offline proxy service suddenly loads a great volume of new data into a database, rendering its memory allocation time much longer. As a result, the overloaded database service fails to process online requests, and many important services, e.g., services for video playing, are affected. A simplified version of the extracted impact subgraph of this incident is presented in Fig. 7. With its help, the root cause is diagnosed as the database service. Moreover, the system is finally recovered by stopping the data loading operations.

[1]More details are in Appendix F.

Real Case II: This is an incident caused by a program bug. The error handling logic of a service (i.e., retrieve service in Fig. 8) is not correctly implemented. As a result, when it receives a bad input with incomplete content, a memory out of bound error happens and the service finally crashes. This incident seriously affects the recommendation function of the system. Fig. 8 presents a simplified version of the extracted impact subgraph of this incident. After operators examined the impact subgraph and identified the root cause of this incident, they rolled back the input to the root cause service and restarted it to recover the system.

B. Limitation of GEM

A limitation of GEM is that it cannot effectively detect and diagnose incidents caused by silent failures [47]. Specifically, only when an incident causes some services to reflect their anomalies in some telemetry data can the incident be detected by GEM, and only when the root cause entity itself reflect its anomalies in some telemetry data can GEM locate the entity as the root cause. When the monitoring infrastructure is badly engineered, or when incidents are caused by faults in some third-party services which are permitted with limited observability, the effectiveness of GEM may degrade.

C. Threats to Validity

The internal threat to validity primarily concerns our implementation of baseline methods. For approaches lacking public implementations [2], [3], [18], we developed our own versions following their published descriptions and using established machine learning toolkits [48]–[50]. System-specific constraints required adaptations of certain baselines. For example, Warden [3] was implemented without operator activity records unavailable in our system. To mitigate this threat, we have carefully implemented and adapted these baselines for testing in our datasets.

The external threat relates to generalizability. While GEM was deployed in the production environment of WeChat, this large-scale system may not represent online systems in other organizations. To mitigate this threat, we have also tested and open-sourced GEM using a publicly available benchmark system and two open-source datasets, facilitating broader adoption of our techniques.

VI. RELATED WORK

Impact Extraction (also denoted as alert aggregation in prior works). Recent research has recognized the challenge of analyzing vast telemetry data anomalies in large-scale systems [23]–[26]. Most approaches perform alert aggregation using textual information, such as AlertSummary [25], which handles alert storms through clustering based on textual and topological similarity. GEM differs from prior approaches in both focus and requirements: (1) GEM prioritizes enhancing incident detection and diagnosis using impact subgraphs aggregated from anomalies, rather than optimizing clustering accuracy; (2) GEM relies solely on machine-generated time series telemetry and system topology, eliminating the need for human-generated textual data.

Incident/Fault Detection. Numerous studies [2]–[6], [9], [15], [51]–[55] have addressed incident/fault detection for online systems. Some focus specifically on fault detection [5], [6], [9], [15], while others target incident detection [2]–[4], [51]–[55]. Among incident detection approaches, some use time series telemetry data [2], [3], while others rely on textual data like logs and user feedback [4], [51]–[55]. GEM belongs to the former category. Among the telemetry data based incident detection methods, Warden [3] is the state of the art, which first prioritizes telemetry through Weighted Mutual Information analysis and perform incident detection using a Balanced Random Forest based classifier. Compared with it, GEM takes the advantage of the extracted impact subgraphs to prioritize incident-indicating telemetry, and further considers relevant service attribute information, allowing more accurate incident detection for service availability assurance.

Incident/Fault Diagnosis. Extensive research [7]–[18], [56] has addressed incident/fault diagnosis, with approaches based either on author-designed heuristics [12]–[18], [56] or supervised learning [7]–[12]. For example, MicroHECL [18] analyzes anomaly propagation chains in service call graphs, while DeepHunt [10] and ART [15] use self-supervised learning on whole-system telemetry. GEM distinguishes itself by: (1) treating impact subgraphs as first-class citizens, eliminating the need to persist all system telemetry for model training; and (2) enabling cold-start diagnosis when incident records are scarce, with gradual evolution through telemetry expansion and parameter optimization.

Model Update Strategies for Software Operation. A recent study [57] also notices the importance of evolving machine learning models for software operation, and performs a comprehensive empirical study on the effectiveness of diverse model update strategies when the data distribution changes (i.e., handling the distribution drifts of $\mathbf{X}$ and $\mathbf{E}$). Compared with it, GEM focuses on incident management, and in addition to applying model re-training to handle distribution drifts, GEM further investigates and proposes solutions for the cold start problem (i.e., the number of samples $|\mathcal{T}|$ for learning is growing from zero), the system scale evolution problem (i.e., the system scale $|\widehat{\mathcal{V}}|$ is growing and may be huge), and the diagnosis telemetry evolution problem (i.e., the number of telemetry $|\mathbf{X}|$ and $|\mathbf{E}|$ for incident diagnosis are gradually incremented). Therefore, the results of GEM complement this prior study [57] on model update strategies for software operation.

VII. CONCLUSION

We propose GEM, a novel framework that extracts and represents issues as system subgraphs, and then use them as first-class citizens in incident management for modern large-scale online systems. GEM by design is evolution-aware, taking the evolution of system scale, data volume and diagnosis telemetry into account for practical incident management. To achieve this, GEM designs lifelong learning techniques on graphs, including graph based models that can incorporate diverse information, a multi-stage criteria for incident detection, and an evolvable PageRank based algorithm for incident diagnosis. We evaluate GEM using both real-world and open-sourced datasets and successfully integrate it into the incident management workflow of WeChat, a real-world production system. Experimental results and real-world deployment confirm the effectiveness of GEM in enhancing incident management practices.

REFERENCES

[1] J. Chen *et al.*, "An empirical investigation of incident triage for online service systems," in *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019, Montreal, QC, Canada, May 25-31, 2019*. IEEE, 2019, pp. 111–120.

[2] Y. Chen *et al.*, "Outage prediction and diagnosis for cloud service systems," in *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, 2019, pp. 2659–2665.

[3] L. Li *et al.*, "Fighting the fog of war: Automated incident detection for cloud systems," in *2021 USENIX Annual Technical Conference, USENIX ATC 2021, July 14-16, 2021*. USENIX Association, 2021, pp. 131–146.

[4] N. Zhao *et al.*, "Real-time incident prediction for online service systems," in *The 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Sacramento, California, United States,November 8-13, 2020*. ACM, 2020.

[5] C. Lee *et al.*, "Heterogeneous anomaly detection for software systems via semi-supervised cross-modal attention," in *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*. IEEE, 2023, pp. 1724–1736.

[6] Z. He *et al.*, "A spatiotemporal deep learning approach for unsupervised anomaly detection in cloud systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 4, pp. 1705–1719, 2023.

[7] S. Zhang *et al.*, "Robust failure diagnosis of microservice system through multimodal data," *IEEE Transactions on Services Computing*, vol. 16, no. 6, pp. 3851–3864, 2023.

[8] Z. Li *et al.*, "Actionable and interpretable fault localization for recurring failures in online service systems," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*. ACM, 2022, pp. 996–1008.

[9] C. Lee *et al.*, "Eadro: An end-to-end troubleshooting framework for microservices on multi-source data," in *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*. IEEE, 2023, pp. 1750–1762.

[10] Y. Sun *et al.*, "Interpretable failure localization for microservice systems based on graph autoencoder," *ACM Transactions on Software Engineering and Methodology*, Sep. 2024, just Accepted.

[11] M. Ma *et al.*, "Automap: Diagnose your microservice-based web applications automatically," in *The Web Conference 2020, WWW 2020, Taipei, Taiwan, April 20-24, 2020*. ACM / IW3C2, 2020, pp. 246–258.

[12] P. Chen *et al.*, "Causeinfer: Automated end-to-end performance diagnosis with hierarchical causality graph in cloud environment," *IEEE Transactions on Services Computing*, vol. 12, no. 2, pp. 214–230, 2019.

[13] M. Kim *et al.*, "Root cause detection in a service-oriented architecture," in *ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '13, Pittsburgh, PA, USA, June 17-21, 2013*. ACM, 2013, pp. 93–104.

[14] J. Lin *et al.*, "Microscope: Pinpoint performance issues with causal graphs in micro-service environments," in *Service-Oriented Computing - 16th International Conference, ICSOC 2018, Hangzhou, China, November 12-15, 2018, Proceedings*, vol. 11236. Springer, 2018, pp. 3–20.

[15] Y. Sun *et al.*, "Art: A unified unsupervised framework for incident management in microservice systems," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, ASE 2024*. New York, NY, USA: Association for Computing Machinery, 2024, p. 1183–1194.

[16] L. Wu *et al.*, "Microrca: Root cause localization of performance issues in microservices," in *IEEE/IFIP Network Operations and Management Symposium, NOMS 2020, Budapest, Hungary, April 20-24, 2020*. IEEE, 2020, pp. 1–9.

[17] G. Yu *et al.*, "Microrank: End-to-end latency issue localization with extended spectrum analysis in microservice environments," in *The Web Conference 2021, WWW 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. ACM / IW3C2, 2021, pp. 3087–3098.

[18] D. Liu *et al.*, "Microhecl: High-efficient root cause localization in large-scale microservice systems," in *43rd IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2021, Madrid, Spain, May 25-28, 2021*. IEEE, 2021, pp. 338–347.

[19] Z. Zeng *et al.*, "A survey on sliding window sketch for network measurement," *Comput. Networks*, vol. 226, p. 109696, 2023.

[20] S. Bhatia *et al.*, "Midas: Microcluster-based detector of anomalies in edge streams," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 3242–3249.

[21] B. Steenwinckel *et al.*, "FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning," *Future Gener. Comput. Syst.*, vol. 116, pp. 30–48, 2021.

[22] L. Tailhardat *et al.*, "NORIA-O: an ontology for anomaly detection and incident management in ICT systems," in *The Semantic Web - 21st International Conference, ESWC 2024, Hersonissos, Crete, Greece, May 26-30, 2024, Proceedings, Part II*, vol. 14665. Springer, 2024, pp. 21–39.

[23] Q. Lin *et al.*, "Log clustering based problem identification for online service systems," in *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016 - Companion Volume*. ACM, 2016, pp. 102–111.

[24] Y. Chen *et al.*, "Identifying linked incidents in large-scale online service systems," in *28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2020, Virtual Event, USA, November 8-13, 2020*. ACM, 2020, pp. 304–314.

[25] N. Zhao *et al.*, "Understanding and handling alert storm for online service systems," in *42nd International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2020, Seoul, South Korea, 27 June - 19 July, 2020*. ACM, 2020, pp. 162–171.

[26] Z. Chen *et al.*, "Graph-based incident aggregation for large-scale online service systems," in *36th IEEE/ACM International Conference on Automated Software Engineering, ASE 2021, Melbourne, Australia, November 15-19, 2021*. IEEE, 2021, pp. 430–442.

[27] H. Zhou *et al.*, "Overload control for scaling wechat microservices," in *Proceedings of the ACM Symposium on Cloud Computing, SoCC 2018, Carlsbad, CA, USA, October 11-13, 2018*. ACM, 2018, pp. 149–161.

[28] Istio. (2024) Istio. [Online]. Available: https://istio.io/

[29] H. Xu *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2018, pp. 187–196.

[30] H. Ren *et al.*, "Time-series anomaly detection service at microsoft," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2019, pp. 3009–3017.

[31] B. Beyer *et al.*, *Site reliability engineering: How Google runs production systems*. " O'Reilly Media, Inc.", 2016.

[32] P. Liu *et al.*, "Fluxrank: A widely-deployable framework to automatically localizing root cause machines for software service failure mitigation," in *30th IEEE International Symposium on Software Reliability Engineering, ISSRE 2019, Berlin, Germany, October 28-31, 2019*. IEEE, 2019, pp. 35–46.

[33] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD 1996, Portland, Oregon, USA*, vol. 96, no. 34. AAAI Press, 1996, pp. 226–231.

[34] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 29–38.

[35] P. Velickovic *et al.*, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[36] Y. Li *et al.*, "Gated graph sequence neural networks," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[37] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[38] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[39] Y. Zhang *et al.*, "Adanpc: Exploring non-parametric classifier for test-time adaptation," in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, vol. 202. PMLR, 2023, pp. 41 647–41 676.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[41] L. Page *et al.*, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[42] A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ser. ICML '04. New York, NY, USA: ACM, 2004, p. 78.

[43] Y. Ozaki *et al.*, "Multiobjective tree-structured parzen estimator for computationally expensive optimization problems," in *Genetic and Evolutionary Computation Conference, GECCO 2020, Cancún Mexico, July 8-12, 2020*. ACM, 2020, pp. 533–541.

[44] Google. (2024) Online boutique: A cloud-native microservices demo application. [Online]. Available: https://github.com/GoogleCloudPlatform/microservices-demo

[45] AIOps Challenge. (2024) Aiops challenge 2021. [Online]. Available: https://competition.aiops-challenge.com/home/competition/1484364004628439089

[46] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *Proceedings of the Twenty-Third International Conference on Machine Learning, ICML 2006, Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, vol. 148. ACM, 2006, pp. 233–240.

[47] C. Lou *et al.*, "Demystifying and checking silent semantic violations in large distributed systems," in *16th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2022, Carlsbad, CA, USA, July 11-13, 2022*. USENIX Association, 2022, pp. 91–107.

[48] G. Lemaître *et al.*, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.

[49] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016, San Francisco, CA, USA, August 13-17, 2016*. ACM, 2016, pp. 785–794.

[50] A. Hagberg *et al.*, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[51] W. Zheng *et al.*, "ifeedback: Exploiting user feedback for real-time issue detection in large-scale online service systems," in *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*. IEEE, 2019, pp. 352–363.

[52] J. Chen *et al.*, "How incidental are the incidents? characterizing and prioritizing incidents for large-scale online service systems," in *35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, Melbourne, Australia, September 21-25, 2020*. IEEE, 2020, pp. 373–384.

[53] J. Gu *et al.*, "Efficient incident identification from multi-dimensional issue reports via meta-heuristic search," in *28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2020, Virtual Event, USA, November 8-13, 2020*. ACM, 2020, pp. 292–303.

[54] S. He *et al.*, "Identifying impactful service system problems via log analysis," in *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*. ACM, 2018, pp. 60–70.

[55] Q. Lin *et al.*, "idice: problem identification for emerging issues," in *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*. ACM, 2016, pp. 214–224.

[56] H. Wang *et al.*, "Groot: An event-graph-based approach for root cause analysis in industrial settings," in *36th IEEE/ACM International Conference on Automated Software Engineering, ASE 2021, Melbourne, Australia, November 15-19, 2021*. IEEE, 2021, pp. 419–429.

[57] Y. Lyu *et al.*, "On the model update strategies for supervised learning in aiops solutions," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 7, Aug. 2024.