Reinforced Neighborhood Selection Guided Multi-Relational Graph Neural Networks

HAO PENG*, Beihang University, China
RUITONG ZHANG, Beihang University, China
YINGTONG DOU, University of Illinois at Chicago, USA
RENYU YANG, University of Leeds, UK
JINGYI ZHANG, Beihang University, China
PHILIP S. YU, University of Illinois at Chicago, USA

Graph Neural Networks (GNNs) have been widely used for the representation learning of various structured graph data, typically through message passing among nodes by aggregating their neighborhood information via different operations. While promising, most existing GNNs oversimplified the complexity and diversity of the edges in the graph, and thus inefficient to cope with ubiquitous heterogeneous graphs, which are typically in the form of multi-relational graph representations. In this paper, we propose RioGNN, a novel **R**einforced, recurs**i**ve and flexible neigh**b**or**o**rhood selection guided multi-relational **G**raph **N**eural **N**etwork architecture, to navigate complexity of neural network structures whilst maintaining relation-dependent representations. We first construct a multi-relational graph, according to the practical task, to reflect the heterogeneity of nodes, edges, attributes and labels. To avoid the embedding over-assimilation among different types of nodes, we employ a label-aware neural similarity measure to ascertain the most similar neighbors based on node attributes. A reinforced relation-aware neighbor selection mechanism is developed to choose the most similar neighbors of a targeting node within a relation before aggregating all neighborhood information from different relations to obtain the eventual node embedding. Particularly, to improve the efficiency of neighbor selecting, we propose a new recursive and scalable reinforcement learning framework with estimable depth and width for different scales of multi-relational graphs. RioGNN can learn more discriminative node embedding with enhanced explainability due to the recognition of individual importance of each relation via the filtering threshold mechanism. Comprehensive experiments on real-world graph data and practical tasks demonstrate the advancements of effectiveness, efficiency and the model explainability, as opposed to other comparative GNN models.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; **Machine learning approaches**; • **Information systems** → **Data mining**; **Web applications**.

Additional Key Words and Phrases: Graph neural network, multi-relational graph, reinforcement learning, node embedding, recursive optimization

*This is the corresponding author.

A preliminary version [15] of this article appeared in the Proceedings of the 29th ACM International Conference on Information and Knowledge Management, Pages 315–324 (CIKM'20). Authors' addresses: H. Peng, School of Cyber Science and Technology, Beihang University, No. 37 Xue Yuan Road, Haidian District, Beijing, 100191, China; email: penghao@buaa.edu.cn; R. Zhang, School of Cyber Science and Technology, Beihang University, No. 37 Xue Yuan Road, Haidian District, Beijing, 100191, China; email: rtzhang@buaa.edu.cn; Y. Dou, Department of Computer Science, University of Illinois at Chicago, Chicago, IL; email: ydou5@uic.edu; R. Yang, School of Computing, University of Leeds, Leeds, LS2 9JT, UK; email: r.yang1@leeds.ac.uk; J. Zhang, School of Cyber Science and Technology, Beihang University, No. 37 Xue Yuan Road, Haidian District, Beijing, 100191, China; email: turuarua@163.com; P. S. Yu, Department of Computer Science, University of Illinois at Chicago, Chicago, IL; email: psyu@uic.edu.

1046-8188/2021/10-ART39 $15.00
https://doi.org/0000001.0000001

ACM Reference Format:
Hao Peng, Ruitong Zhang, Yingtong Dou, Renyu Yang, Jingyi Zhang, and Philip S. Yu. 2021. Reinforced Neighborhood Selection Guided Multi-Relational Graph Neural Networks. *ACM Transactions on Information Systems* 1, 1, Article 39 (October 2021), 47 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

The advancement of Graph Neural Networks (GNNs) enables the effective representation learning for a variety of areas [104] including bioinformatics, chemoinformatics, social networks, natural language processing [76, 78], social events [5, 75, 77], recommender system [83], spatial-temporal traffic [30, 80], computer vision and physics [3] where graphs are primarily the denotation. GNN models are proved to reach the performance target over massive datasets – citation networks [81, 91], biochemical networks [74, 123], social networks [5, 19], knowledge graph [58, 85, 109], commodity networks [79], API call networks [37], etc. – on different tasks, such as node classification [12, 32, 49, 73, 96], node clustering [43, 72], link prediction [48, 52, 114], graph classification [18, 74, 92, 111], etc. At the core of GNNs is to operate various aggregate functions [32, 49, 96, 107] on the graph structure by passing node features to the neighbors; each node aggregates the feature vectors of its neighbors for computing and updating its new feature vector. Empirically, iterations of aggregation or message passing come into a node embedding vector – a numerical capture of both structural information within the node's multi-hop neighborhood and the attribute information – empowered by the label propagation mechanisms [29, 57, 122].

Heterogeneous graphs are ubiquitous in real-world systems; a graph typically consists of nodes with multiple types and multi-relational edges between nodes. For example, in Yelp spam review data [84], there are heterogeneous nodes (e.g., businesses, reviews, users, etc.) and relations (e.g., posted by the same user, under the same product with the same star rating, and under the same product posted in the same month between two reviews). However, existing iterative aggregation mechanisms of GNNs have yet to elaborately consider the diversity of semantic relations and the usability of the proposed models. Homogeneous GNNs such as GraphSAGE [32], GCN [49], GAT [96], GIN [107] ignore or simplify the diversity and complexity of the nodes and edges in practical networks, which is inadequate to represent the heterogeneity of data. To solve the above problem, relational GNNs [71, 85, 121] are proposed but fail to capture multiple hop or complex relations. Sampling-based heterogeneous GNNs guided by hand-crafted meta-paths [101], meta-graphs [108] and meta-schema [36] are solely based on data types and their structured connections. This drawback substantially impedes the generalization of such heterogeneous GNNs in practical fine-grained tasks – e.g., fraud detection [1, 14], disease diagnosis [20], etc. – where it is infeasible or inefficient to externalize the inherent entity relations through meta-structures such as meta-path, meta-graph and meta-schema. Take the detection and diagnosis of diabetes and its suspected diseases, based on the MIMIC-III dataset [44], as an example. Observably, a portion of patients with diabetes tends to have symptoms that cause glaucoma, while glaucoma patients do not often experience issues in blood sugar, insulin and other test indicators. Accordingly, one can easily define explicit relations such as *having hyperglycemia score in the blood test*, *having high proteinuria scores in urine test*, *having symptoms of glaucoma in vision test*, *having high intraocular in pressure test*, etc., between any two patients. It is far more useful to specify relations based on common attributes shared by entities, and less dependent upon strict entity connections as opposed to the meta-structure based approaches which have to leverage complicated automated generation technology [68] or manual experience [6, 39, 62]. Hence, it is more effective to explore and exploit the explicit relations, stemming from task-specific characteristics, for carrying out downstream applications.

In an attempt to extend GNNs for supporting heterogeneous graph embedding, many approaches rely on a combination of sophisticated neural networks [100]. For instance, HetGNN [113] aggregates multi-modal features from heterogeneous neighbors by combining bi-LSTM, self-attention, and types combination. RSHN [121] utilizes coarsened line graph neural network (CL-GNN) along with the message passing neural network (MPNN) to learn

node and edge type embedding simultaneously. HGT [41] leverages type dependent parameters based mutual attention, message passing, residual connection, target-specific aggregation function, etc. MAGNN [22] makes use of meta-path sampling, intra-meta-path and inter-meta-path aggregation technologies to embed a node with the targeted type. Nevertheless, they lack the analysis of more practical or fine-grained application tasks and require strong domain knowledge to build the complex neural network structures. The usability of the proposed GNN model should also be designed in a more convenient way.

To this end, we propose RioGNN, a novel Reinforced, recursIve and scalable neighbOrhood selection guided multi-relational Graph Neural Network, to navigate complexity of customized neural network structures whilst maintaining relation-dependent representations. For domain task driven graph representation learning, we introduce *multi-relational graph* to reflect the heterogeneity of nodes, edges, attributes and labels. Herein, a relation is referred to as a specific type of edge between two nodes, connected with each other through explicit common attributes or implicit semantics, e.g., two products released in the same month, two movies directed by the same director, etc. Departing from heterogeneous information network (HIN) [87], the multi-relational graph is able to flexibly characterize and explicitly differentiate the edge types without the need for specifying semantic connectivity between any two nodes strictly following entity-associated meta-structures. For a given relation, we can conduct the sampling procedure upon the original graph for extracting neighbors for each node in the graph.

To diminish the complexity of neural network units, RioGNN optimizes the process of neighbor selection when aggregating neighbor information for a center node embedding. To avoid the embedding over-assimilation among different types of nodes, we first employ a label-aware neural similarity measure to ascertain the most similar neighbors based on node attributes. Particularly, this is achieved by a neural classifier that transforms the supervised signals (e.g., high-fidelity annotated labels) and original node features to calculate the node similarity. To follow up, we carry out a relation-aware neighbor selection to choose the most similar neighbors of a targeting node within a reinforced relation before aggregating all neighborhood information from different relations to obtain the eventual node embedding. To improve the neural classifier training efficiency, we optimize the filtering threshold within each relation through RSRL, a novel Recursive and Scalable Reinforcement Learning framework with estimatable depth and width for different scales of a heterogeneous graph in a recursive manner. Specifically, we exploit two general relation-aware RSRL methods – using both discrete and continuous strategies – for pinpointing the optimal number of neighbors of different relations. The discrete and continuous approaches can generally provide more choices in the face of different datasets and application scenarios. RSRL not only facilitates to learn discriminative node embeddings, but also makes the model more explainable as we can recognize the individual importance of each relation via the filtering thresholds. RSRL-based relation-aware neighbor selector can be integrated with any mainstream reinforcement learning models [31, 35, 50, 86, 103] and neighbor aggregation functions [32, 96] used for specific scenarios.

We integrate the aforementioned techniques with the vanilla GNN as a layer of RioGNN and devise multi-layered RioGNN to learn high-order node representations according to the specific requirements of downstream tasks. This paper mainly targets those tasks with node-level embedding and learns the multi-relation node representation in a semi-supervised manner. We evaluate the effectiveness, efficiency and explainability of RioGNN by applying it to two tasks of fraud detection and diabetes detection, using Yelp, Amazon and MIMIC-III datasets. Experiments assess how RioGNN underpins downstream tasks including transductive node-classification, inductive node-classification and node clustering. Results show that RioGNN significantly improves various downstream tasks over state-of-the-art GNNs as well as dedicated heterogeneous models by 0.70%–32.78%. We show that our RSRL framework not only boosts the learning time by up to 4.52x, but also achieves 4.90% improvement in node classification. We also evaluate the sensitivity of RioGNN to hyper-parameters in the above tasks. Finally, we carry out a series of case studies to showcase how RSRL automatically learns the importance and engagement of implicit relations in different tasks. The source code and datasets are publicly available at https://github.com/safe-graph/RioGNN.

The contributions of this work are summarized as follows:

- The first task-driven GNN framework based on multi-relational graphs, making the best use of relational sampling, message passing, metric learning and reinforcement learning to guide neighbor selection within and across different relations.
- A flexible neighborhood selection framework that employs a reinforced relation-aware neighbor selector with label-aware neural similarity neighbor measures.
- A recursive and scalable reinforcement learning framework that learns the optimized filtering thresholds via estimable depth and width for different scales of graphs or tasks.
- The first study on the explainability of multi-relational GNNs from the perspective of importances of different relations.

We expand upon our preliminary work [15], by extending CAmouflage-REsistant GNN (CARE-GNN) model exclusively for fraud detectors against camouflaged fraudsters to a more general architecture underpinning a wide range of practical tasks. Specifically, the improvements encompass: 1) giving a full version of definition, motivation and aim of multiple relation graph neural networks under different practical tasks; expanding the label-aware similarity neighbor measure from one layer to multiple layers to select the similar neighbors; 2) proposing a novel recursive and scalable reinforcement learning framework to optimize the filtering threshold for each relation along with the GNN training process in a general and efficient manner, instead of the previous Bernoulli Multi-armed Bandit method; 3) leveraging both discrete and continuous strategies to find the optimal neighbors of different relations to be selected under the reinforcement learning framework; 4) carrying out extensive experiments on three representative and general-purpose datasets, not limited to the fraud detection scenario. Furthermore, more in-depth experimental results are discussed to demonstrate the effectiveness and efficiency of the proposed architecture. We supply the variances of the results of multi-relational graph representation learning. We also showcase the explanation of importances of different relations from a new perspective based on the filtering threshold of the proposed RSRL framework.

The paper is structured as follows: Section 2 outlines the preliminaries and the problem formulation, and Section 3 describes the technical details involved in RioGNN. Experimental setup and results are discussed in Section 4 and Section 5, respectively. Section 7 presents the related work before we conclude the paper in Section 8.

2 BACKGROUND AND OVERVIEW

2.1 Problem Definition

In this section, we firstly define the multi-relational graph and multi-relational GNN. All important notations in this paper are summarized in Table 1.

Definition 2.1. Multi-Relational Graph (MR-Graph). MR-Graph is defined as $\mathcal{G} = \left\{\mathcal{V}, \mathcal{X}, \{\mathcal{E}_r\}|_{r=1}^{R}, Y\right\}$, where $\mathcal{V}$ is the set of nodes $\{v_1, \ldots, v_n\}$, and n is the number of nodes in the graph. Each node v_i has a d-dimensional feature vector $\mathbf{x}_i \in \mathbb{R}^d$ and $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ represents a set of all node features. $e_{i,j}^r = (v_i, v_j) \in \mathcal{E}_r$ is an edge between v_i and v_j with a relation $r \in \{1, \cdots, R\}$, where R is the number of relations. Note that an edge can be associated with multiple relations, and there are R different types of relations. Y is the set of labels for each node in $\mathcal{V}$.

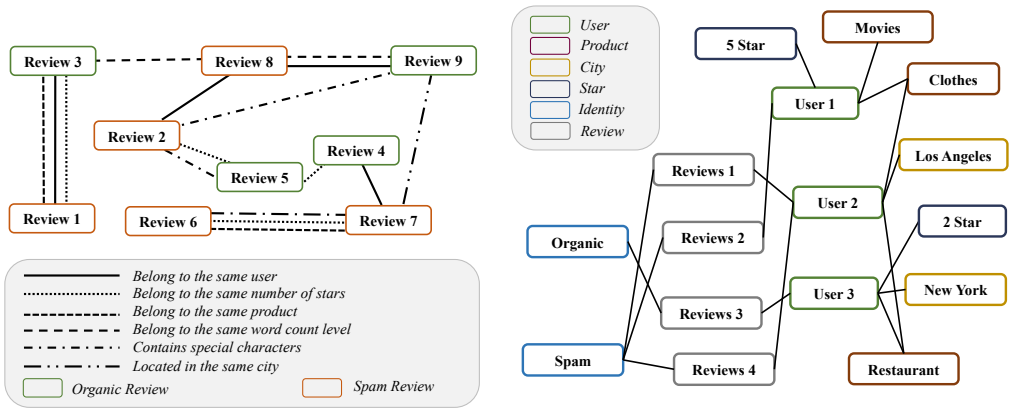
The multi-relational graph directly uses the elements to be classified as nodes, and the key relations of elements with different labels are used as multiple connections, which can be widely used in challenging classification tasks. It is worth noting that, departing from HIN, the multi-relational graph is able to flexibly characterize and explicitly differentiate the edge types without the need for specifying semantic connectivity between any

two nodes strictly following entity-associated meta-structures. We exemplify its applicability by two real-world applications and compare the difference between the MR-Graph based modeling and the HIN-based approach:

Spam Review Detection. Spam reviews are referred to as those fabricated reviews posted to products or merchants with the intent of promoting their targets. Fraud detection has to identify spam reviews from organic ones. As spam comments add some special characters or simulate benign email behaviors (such as one user who posts spam emails while maintaining a certain frequency of organic comments) to avoid being found out, this brings challenges to distinguishing spam comments. We consider the comments with different labels as nodes, and different representative interactions as different types of connections to build a multi-relational graph, thereby transforming this problem into a two-classification problem. As shown in Figure 1(a), an MR-Graph example depicts the organic reviews, spam reviews and their interactions extracted from the e-commerce review data [70, 84]. We extracted representative interactions between two reviews that are closely associated with the fraudulent behavior, and represented them as different types of edges – *Belonging to the same user*, *Having the same star rating*, *Targeting the same product posted in the same month*, *Belonging to the same word count*

Table 1. Notations.

Symbol	Definition
$\mathcal{G};\mathcal{V};\mathcal{E};\mathcal{X}$	Graph; Node set; Edge set; Node feature set
$y_v; Y$	Label for node v; Node label set
$r; R$	Relation; Total number of relations
$l; L$	GNN layer number; Total number of layers
$b; B$	Training batch number; Total number of batches
$e; E$	Training epoch number; Total number of epochs
$\mathcal{V}_{train}; \mathcal{V}_b$	Nodes in the training set; Node set at batch b
$\mathcal{E}_r^{(l)}$	Edge set under relation r at the l-th layer
$\mathbf{h}_v^{(l)}$	The embedding of node v at the l-th layer
$\mathbf{h}_{v,r}^{(l)}$	The embedding of node v under relation r at the l-th layer
$\mathcal{D}^{(l)}(v, v')$	The distance between node v and v' at the l-th layer
$S^{(l)}(v, v')$	The similarity between node v and v' at the l-th layer
$p_r^{(l)} \in P$	The filtering threshold for relation r at the l-th layer
$RLF^{(l)}$	The Reinforcement Learning Forest at the l-th layer
$RLT_r^{(l)}$	The Reinforcement Learning Tree for relation r at the l-th layer
$RL_r^{(l)(d)}$	The Reinforcement Learning Module for relation r at the l-th layer in d-th depth
$W_r^{(l)(h)}$	The width of Reinforcement Learning Tree for relation r at the l-th layer in d-th depth
$D_r^{(l)}$	The depth of Reinforcement Learning Tree for relation r at the l-th layer
$a_r^{(l)} \in A$	RL action space;
$s(\mathcal{D}_r^{(l)(d)})^{(e)}$	RL state for relation r at the l-th layer in d-th depth when the epoch is e
$g_r^{(l)(d)(e)}$	RL reward for relation r at the l-th layer in d-th depth when the epoch is e
$f_r^{(l)(d)(e)}$	RL iterative function for relation r at the l-th layer in d-th depth when the epoch is e
$\mathcal{N}_r^l(v)$	Nodeset after RL filtering for relation r at the l-th layer
$\text{AGG}_r^{(l)}$	Intra-relation aggregator for relation r at the l-th layer
$\text{AGG}_{all}^{(l)}$	Inter-relation aggregator at the l-th layer
$\mathbf{z}_v$	Final embedding for node v

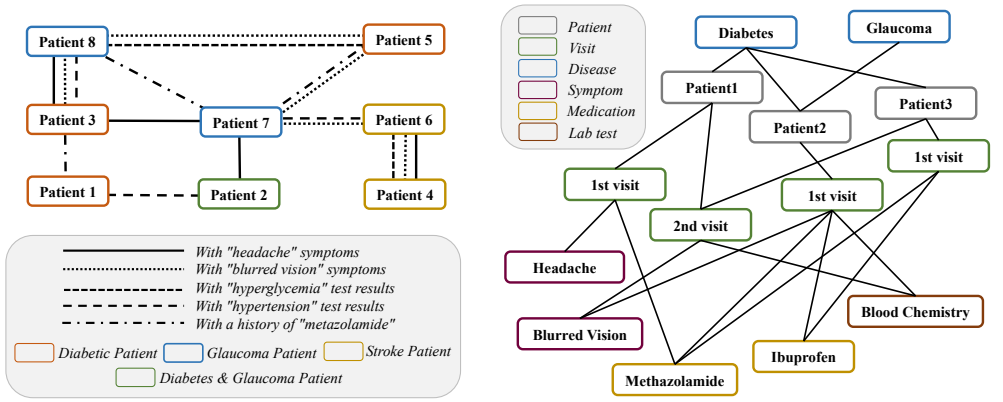


Relational Fraud Detection Graph | **Heterogeneous Fraud Detection Graph**

(a) An MR-Graph example for fraud review detection.

(b) A HIN example from review data [4, 93].

Fig. 1. Graph Modeling in Fraud Review Detection.



Relational Disease Diagnosis Graph | **Heterogeneous Disease Diagnosis Graph**

(a) An MR-Graph for diabetes and disease detection.

(b) An example of heterogeneous Electronic Health Records (EHR) graph, reproduced from [62].

Fig. 2. Graph Modeling in Disease Diagnosis.

level, *Containing special characters* and *Targeting products located in the same city*. As an alternative, traditional HIN-based modeling (Figure 1(b)) pays more attention to the relations outlined by structured connections.

Disease Diagnosis. In the task of disease diagnosis, diabetes, stroke, and glaucoma are common diseases in middle-aged and elderly people, and their early symptom recognition is very important. However, because these three diseases have similar symptoms, it is difficult to distinguish patients in clinical practice. For example, the symptoms of stroke include loss of vision, sudden weakness and tingling sensations, which are similar to symptoms in patients with type II diabetes. In addition, one of the early symptoms of diabetes is blurred vision

caused by changing fluid levels. Therefore, the eyes may change shape, disturbing the focusing ability of the eyes. Although this visual impairment may indicate diabetes, it is also true in glaucoma patients. Here, taking the patient as the node of the multi-relational graph and connecting the patients with different similar symptoms into different types of edges can convert the task into a multi-classification task. Figure 2(a) illustrates an MR-graph of patients for disease diagnosis. We believe to model and represent the relationship between the following types of patients: *with "headache" symptoms*, *with "blurred vision" symptoms*, *with "hyperglycemia" test results*, *with "hypertension" test results*, *with a history of "metazolamide"*, etc. are more helpful for the diagnosis of diabetes and its suspected diseases of patients. Even, there may be multiple relationships between two patients. For example, *Patient 5* and *Patient 8* have two relationships of *with "blurred vision" symptoms* and *with "hypertension" test results* in common. Meanwhile, previous HIN-based Electronic Health Records (EHR) modelings [6, 39, 62] focused on the correlation and fusion of different attributes or types of data, and the corresponding methods are suitable for the diagnosis of all kinds of diseases. Moreover, there is a lack of fine-grained analysis of certain definite diseases. We also give an illustration of the heterogeneous Electronic Health Records graph in Figure 2(b).

Definition 2.2. Multi-Relational GNN. Graph neural network (GNN) is a deep learning framework to embed graph-structured data via aggregating the information from its neighboring nodes [25, 32, 49, 96]. Based on Definition 2.1, we can then outline a unified formulation of the Multi-Relational GNN from the perspective of multi-layer neighbor aggregation according to different relations. For a central node v, the hidden or aggregated embedding of node v at l-th layer is referred to as $\mathbf{h}_v^{(l)}$:

$$\mathbf{h}_v^{(l)} = \sigma(\mathbf{h}_v^{(l-1)} \oplus AGG^{(l)}(\{\mathbf{h}_{v',r}^{(l-1)} : (v, v') \in \mathcal{E}_r^{(l)}\}|_{r=1}^{R})), \tag{1}$$

where $\mathcal{E}_r^{(l)}$ denotes the edges under the relation r at the l-th layer, and $\mathbf{h}_{v',r}^{(l-1)}$ indicates to the aggregated embedding of neighboring node v' under relation r. AGG denotes the aggregation function that maps the neighborhood information from different relations into a vector, e.g., mean aggregation [32] and attention aggregation [96]. $\oplus$ is the operator that combines the information of node v and its neighboring information through either concatenation or summation [32]. We initialize the node embedding $\mathbf{h}_v^{(0)}$ with the input d-dimensional feature vector x. The GNN is trained with partially labeled nodes with binary classification loss functions. Instead of directly aggregating the neighbors for all relations, we separate the aggregation part as *intra-relation* aggregation and *inter-relation* aggregation process. During the intra-relation aggregation process, the embedding of neighbors under each relation is aggregated simultaneously. Then, the embeddings for each relation are combined during the inter-relation aggregation process. Finally, the node embeddings at the last layer are used for predictions.

2.2 Problem Scope and Challenges

In practical applications, we model multi-relation graphs, and take actual problems as node classification tasks in a semi-supervised learning manner. After constructing a multi-relational graph according to domain knowledge, e.g., Spam Review Detection in Figure 1(a), Disease Diagnosis in Figure 2(a), etc., a multi-relational GNN can be trained. However, when we train more discriminative, effective and explainable node embedding, there are three main challenges facing the Multi-Relational GNNs:

How to cope with misbehaved nodes during neighbor aggregation in GNNs (Challenge 1). The input node features $\mathcal{X}$, often extracted based on heuristic methods such as TF-IDF, Bag-of-Words, Doc2Vec, etc., are susceptible to such misbehavior as adversarial attacks, camouflages [15, 89], or simply imprecise feature selection. Consequently, the numerical embedding of a central node tends to be assimilated by misbehaved neighboring nodes. For instance, in the spam review detection task, adversarial or camouflaged behaviors are non-negligible noises that drastically reduce the accuracy of feature representation learning by GNNs. Either feature [16, 53] or relational [46, 119] camouflages could similarize the features of misbehaved and benign entities, and further mislead GNNs to generate uninformative node embeddings. In the medical disease diagnosis task, textual attribute

based feature selection may not be able to extract high-level or fine-grained semantics, and thus easily lead to imprecise node characteristics. Hence, these issues necessitate an effective similarity measure to filter the neighbors before applying into any GNNs.

How to adaptively select the most suitable neighbor nodes based on the similarity measure (Challenge 2). Data annotation is expensive for most practical problems, and we cannot select all similar neighbors under each relationship through data labeling. The method of directly regarding the filtering threshold as a hyper-parameter [61, 112] is no longer valid for multiple relationship graphs with numerous noisy or misbehaved nodes. First, different relationships have different feature similarity and label similarity. Secondly, different relationships have different precision requirements for the filtering threshold. Therefore, an adaptive sampling mechanism must be designed so that the optimal number of similar neighbors can be selected for specific relationship requirements in a dynamic environment.

How to efficiently learn and optimize the filtering threshold in a continuous manner (Challenge 3). Our preliminary work [15] adopts the Bernoulli multi-armed bandit framework [94] with a fixed strategy to strengthen the learning of the filtering threshold. However, it is substantially limited by the observation range of the state and manually-specified strategies, and hence the final convergence result of the filtering threshold tends to be locally optimal. In addition, for maintaining the prediction accuracy, in the face of large-scale datasets, it is imperative to reduce the adjustment step size of the filtering threshold or use continuous action space. This procedure will undoubtedly expand the action space, leading to an increased number of convergence periods and a huge growth of calculation, possibly with a loss of accuracy. This issue therefore necessitates an automatic and efficient reinforcement learning framework that can quickly obtain sufficient and high-quality solutions.

3 METHODOLOGY

Figure 3 depicts the RioGNN's overall architecture consisting of three key modules – label-aware similarity measurement (Section 3.1), similarity-aware neighbor selector (Section 3.2), and relation-aware neighbor aggregator (Section 3.3). In addition, we describe the overall algorithm and optimization in Section 3.4.

3.1 Label-aware Neural Similarity Measure

Compared with unsupervised similarity metrics like Cosine Similarity [110] or Neural Networks [110], many practical problems like financial fraud, disease diagnosis, etc., require extra domain knowledge (e.g., high-fidelity data annotations) to identify anomaly instances. To this end, we design a parameterized node similarity measure, i.e., label-aware neural similarity measure, using supervision signals from domain experts. AGCN [54] employs a Mahalanobis distance plus a Gaussian kernel, while DIAL-GNN [10] adopts the parameterized cosine similarity. NSN [56] unitizes bilinear similarity based inner product and hyperspherical learning strategies. However, all of them have non-negligible time complexity $O(\bar{k}d)$, where $\bar{k}$ denotes the average degree of nodes, often very high in real-world graphs, and d represents the feature dimension. This leads to a loss of efficiency when discriminating the node representation learning.

Inspired by GraphMix [97], with a combination of Fully-Connected Network (FCN) and linear regularization, we adopt an FCN as the node label predictor at each layer of RioGNN, and use the l_1-distance between the prediction results of two nodes as the measure of the in-between similarity. It is a simple and efficient regularizer for semi-supervised node classification using GNNs. At the l-th layer, when calculating the distance between one intermediate node v and one of its neighbors v' under relation r, i.e., the edge $(v, v') \in \mathcal{E}_r$, we take their embedding in the previous layer $\mathbf{h}^{(l-1)}$ as input, and apply the non-linear activation function σ (we use $tanh$ in our work). The distance between v and v' is the l_1-distance of two embeddings:

$$\mathcal{D}^{(l)}(v, v') = ||\sigma(FCN^{(l)}\mathbf{h}_v^{(l-1)}) - \sigma(FCN^{(l)}\mathbf{h}_{v'}^{(l-1)})||_1. \tag{2}$$

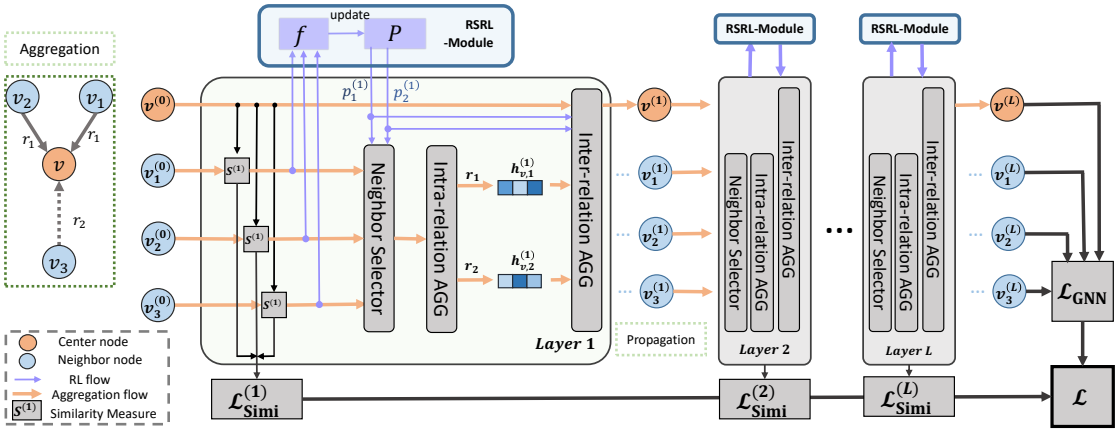


Fig. 3. RioGNN architecture.

Thus, the similarity of the two nodes can be defined as:

$$S^{(l)}(v, v') = 1 - \mathcal{D}^{(l)}(v, v'). \tag{3}$$

The time complexity of our approach can be reduced from $O(\bar{k}d)$ to $O(d)$. In general, the computational cost is low because for each node in the node set $\mathcal{V}$, we do not use the combined embedding of its k neighbors with d-dimensional features to measure similarity like LAGCN [7], but only consider the label predicted by FCN based on its own feature.

To train similarity measure with a direct supervision signal from the labels, we define the cross entropy loss of the FCN in the l-layer as:

$$\mathcal{L}_{Simi}^{(l)} = \sum_{v \in \mathcal{V}} -log(y_v \cdot \sigma(FCN^{(l)}(\mathbf{h}_v^{(l)}))). \tag{4}$$

Further, we define the cross entropy loss of label-aware similarity measure for the entire network as:

$$\mathcal{L}_{Simi} = \sum_{l=1}^{L} \mathcal{L}_{Simi}^{(l)}. \tag{5}$$

During the training process, the similarity measure parameters of FCNs are directly updated through the loss function. It ensures that similar neighbors can be quickly selected within a few batches and facilitates to regularize the GNN training process.

In this subsection, we propose a label-aware similarity detection method for the first challenge in Sec. 2.2. This method is based on node labels to effectively avoid interference caused by bad node camouflage in actual scenes, and reduces the complexity of similarity, which provides a stable basis for subsequent neighbor filtering.

3.2 Similarity-aware Adaptive Neighbor Selector

To select appropriate neighbors adaptively, we design a similarity-aware neighbor selector to filter misbehaved nodes stemming from adversarial behaviors or inaccurate feature extraction. More specifically, for each central node, the selector utilizes *Top-p Sampling* along with adaptive filtering thresholds to construct similar neighbors under each relation. Since the filter thresholds for different relations at different layers tend to be dynamically updated during the training phase, we propose RSRL, a recursive and scalable reinforcement learning framework to optimize the filtering threshold for each relation in an efficient manner.

3.2.1 Top-p Sampling. Before aggregating the information from both central node v and its neighborhood, we perform a *Top-p sampling* to filter the dissimilar neighbors according to different relations. A filtering threshold $p_r^l \in [0, 1]$ for relation r at the l-th layer indicates the selection ratio from all neighbors. For instance, all neighbor nodes under the relation r are retained if p_r is 1. More specifically, during the training phase, for a node v in one batch under the relation r, we first calculate a set of similarity scores $\{S^l(v, v')\}$ by using Eq. 3 at l-th layer where the edge $(v, v') \in \mathcal{E}_r^l$. We then rank the neighbors of each central node v in descending order, based on $\{S^l(v, v')\}$, and take the top $p_r^l \cdot |\{S^l(v, v')\}|$ neighbors as the selected ones, i.e., $\mathcal{N}_r^l(v)$, at the l-th layer. The residual nodes will be discarded at the current batch and not attend the following aggregation process within the layer.

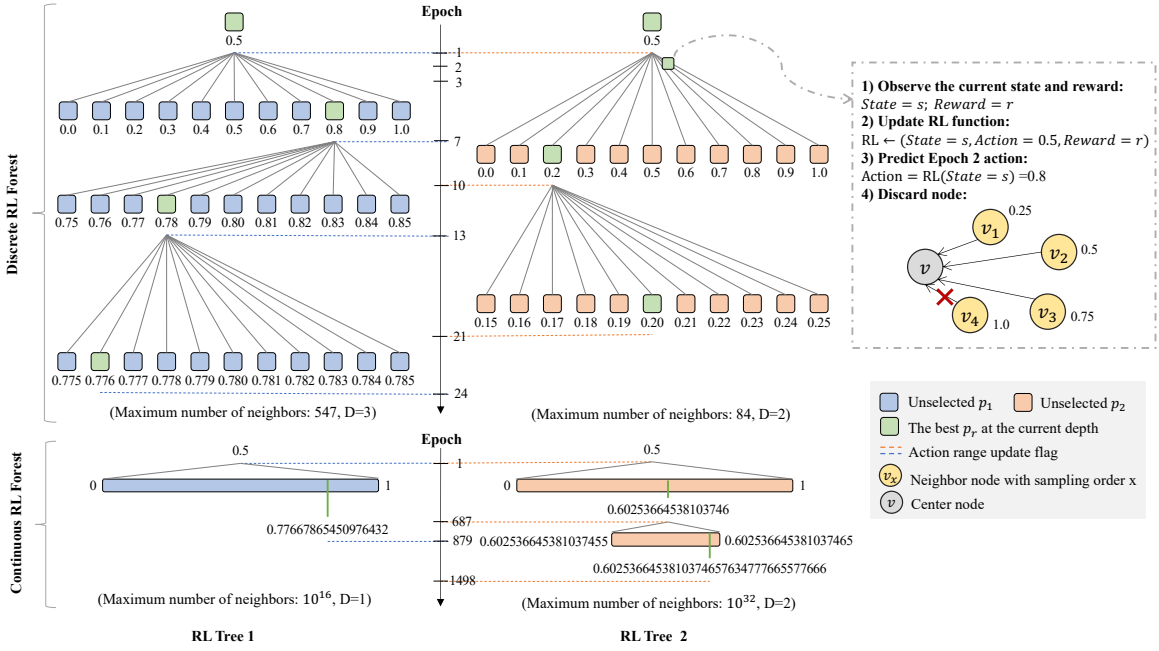


Fig. 4. One layer Reinforcement Learning Forest.

3.2.2 RSRL Framework. Previous work [61, 112] regards the filtering threshold as a hyper-parameter, which is no longer valid for multi-relational graph with numerous noisy or misbehaved nodes. To solve this, our preliminary work [15] adopted the Bernoulli multi-armed bandit framework [94] with a fixed learning strategy and dynamically updated the filtering threshold. However, the effectiveness of this approach is largely impeded

by the limited observation range of states and the manually-specified strategy. As a result, the final convergence outcome of the filtering threshold tends to be local optimal. In the face of larger-scale datasets, the maintenance of the prediction accuracy also needs to reduce the adjustment step size of the filtering threshold. This process will increase the number of convergence epochs, and bring in an increase in the amount of calculation and a loss of accuracy.

Overview. To address these problems, we propose a novel Recursive and Scalable Reinforcement Learning framework RSRL, upon traditional Reinforcement Learning based approaches [21, 51, 63], to not only update strategies through the learning environment but also the recursive structure can be used to quickly and accurately meet the accuracy requirements of different relations. Figure. 4 depicts the forest-based learning architecture. The specific process of a tree in each epoch is shown on the right, where s and r respectively represent the state and reward after the previous epoch, and a (the example in the figure is 0.5) represents the predicted action in the current epoch. We formulate RSRL as an L-layer Reinforcement Learning (RL) Forest, and define the l-th layer forest as:

$$RLF^{(l)} = \{RLT_r^{(l)}\}|_{r=1}^{R} = \{\{RL_r^{(l)(d)}\}|_{d=1}^{D_r^{(l)}}\}|_{r=1}^{R}, \tag{6}$$

$RLF^{(l)}$ actually indicates the process of obtaining the best relational filtering threshold combination at the l-th layer. Each relation independently constructs a RL Tree $RLT_r^{(l)}$ with an adaptive depth $D_r^{(l)} = \lceil \log_\alpha k_r \rceil$ and a width $W_r^{(l)(d)} = \frac{1}{\alpha^d}$. α is the weight parameter of depth first and breadth first, and k_r is the maximum number of neighbors contained in the node in relation r. $RLT_r^{(l)}$ performs a Reinforcement Learning $RL_r^{(l)(d)}$ for filtering the threshold with an accuracy $W_r^{(l)(d)}$ at each depth. At the l-th layer, $RLT_r^{(l)}$ acquires the best filtering threshold $p_r^{(l)(d)}$ of neighbor nodes with higher accuracy than the previous depth of relation r through multiple RL recursively, until the threshold for maximum accuracy requirements is found at the depth $D_r^{(l)}$.

The $RLT_r^{(l)}$ recursive process is expressed as:

$$p_r^{(l)(d)} \xleftarrow{RL_r^{(l)(d)}} \{p_r^{(l)(d-1)} - \frac{W_r^{(l)(d)}}{2}, p_r^{(l)(d-1)} + \frac{W_r^{(l)(d)}}{2}\}. \tag{7}$$

where $p_r^{(l)(d)}$ represents the optimal proportion of neighbor nodes in the relation r to be discarded when the depth of the RL tree is d in the l-th layer. The learning range of the RL module of each depth is the value within $\pm\frac{W_r^{(l)}}{2}$ of the filter threshold $p_r^{(l)(d-1)}$ selected by the previous depth. When the recursive process reaches the maximum depth $D_r^{(l)}$, we obtain the final filtering threshold $p_r^{(l)}$ of the relation r in the l-th layer. Considering that the complexity has a linear relationship with the size of the action space [17], this process carries out a precision recursion on RL actions and can reduce the time from $O(k_r)$ to $O(\alpha \log_\alpha k_r)$, where $\alpha \in (1, k_r)$ and k_r is the maximum node degree under relation r.

Details of RL Process. We express an RL module as a Markov Decision Process $MDP < A, S, R, F >$ for the filtering threshold of a relation. A and S are the action space and state space, respectively; R is the reward function, and F is the iteration functions and termination conditions. To better deal with datasets with different sizes and diverse scenarios, we break down our solution into two distinct categories: Discrete Reinforcement Learning (D-RL) and Continuous Reinforcement Learning (C-RL).

- **Action:** We define the action space A of $RL_r^{(l)(d)}$ by collecting all actions $a_r^{(l)(d)} \in \{p_r^{(l)(d-1)} - \frac{W_r^{(l)(d)}}{2}, p_r^{(l)(d-1)} + \frac{W_r^{(l)(d)}}{2}\}$ when the relation r is at the depth d of the l-th layer. The discrete scheme D-RL divides the action space into α discrete actions on average. In the continuous scheme C-RL, the action space in d-th depth is a continuous floating point number with the width of $W_r^{(l)(d)}$. The compatibility of the two types of action space can effectively adapt to a variety of reinforcement learning algorithms. Due to the low precision

requirements of the filtering threshold for small and medium-scale datasets, discretization of actions can reduce the number of action explorations [47] while meeting the basic accuracy requirements, which ensures efficient access to high-performance areas. For the dataset with large-scale neighbors, a large number of discrete actions that meet the high-precision requirements will affect the learning effect [17]. We propose to generalize the filtering threshold to the continuous action space to improve the accuracy of large-scale data sets by reducing the spatial range multiple times.

- **State:** Since it is impossible to directly perceive the classification loss of GNN as the environment state, we calculate the average node distance of each epoch as the state through the distance measure of label perception (Eq. (2)). In the l-th layer of the e-th epoch, the state s of relation r in the d-th depth is:

$$s_r^{(l)(d)(e)} = \frac{\sum_{(v,v')\in\mathcal{E}_r^{(l)(d)(e)} \mathcal{D}^{(l)}(v,v')^{(e)}}{|\mathcal{E}_r^{(l)(d)(e)}|}, \tag{8}$$

where $\mathcal{E}_r^{(l)(d)(e)}$ is the set of edges filtered in the l-th layer and d-th depth of the e-th epoch under relation r.

- **Reward:** For each relation, the goal is to ascertain a filtering threshold $p_r^{(l)(d)}$ so that the selected neighbor node and the central node are as close as possible. We therefore use the similarity (Eq. (3)) as the decisive factor within the reward function. In the l-th layer of the e-th epoch, the reward g of relation r to d-th depth is:

$$g_r^{(l)(d)(e)} = \tau \cdot (\frac{\sum_{(v,v')\in\mathcal{E}_r^{(l)(d)(e)} \mathcal{S}^{(l)}(v,v')^{(e)}}{|\mathcal{E}_r^{(l)(d)(e)}|}), \tag{9}$$

where τ is the weight parameter, and the meaning of $\mathcal{E}_r^{(l)(d)(e)}$ is the same as the definition in the state.

- **Iteration and Termination:** Before starting each epoch, RL observes the state $s_r^{(l)(d)(e)}$ of the environment after the previous epoch of action $a_r^{(l)(d)(e-1)}$ and obtains a reward $g_r^{(l)(d)(e-1)}$. They are then used to update the iterative function (broadly refers to the function of strategy iteration or value iteration process of reinforcement learning) f. The iterative function of each $RL_r^{(l)(d)}$ is as follows:

$$f_r^{(l)(d)} \leftarrow s_r^{(l)(d)(e-1)}, s_r^{(l)(d)(e)}, a_r^{(l)(d)(e-1)}, g_r^{(l)(d)(e-1)}. \tag{10}$$

Then we can use the iterative function to predict the action a from the current state s, which is the filtering threshold:

$$p_r^{(l)(d)(e)} = a_r^{(l)(d)(e)} = f_r^{(l)(d)}(s_r^{(l)(d)(e)}). \tag{11}$$

Here, for the output of the action, the activation function of the classification type is used to represent them in D-RL, such as softmax. And in C-RL, we use the activation function of the return value type to represent them, such as *tanh*. Since what we propose is a general framework applicable to a variety of reinforcement learning algorithms, the specific definition of the iterative function depends on the actual algorithm. We test the applicability of the RSRL framework to various mainstream reinforcement learning algorithms in Sec. 5.3. To improve the equalization efficiency, we assume the RL will be terminated as long as the same action appears three times in a row at the current accuracy $W_r^{(l)(d)}$. Specifically, in the l-th layer of the e-th epoch, the termination conditions of relation r in the d-th depth are defined as follows:

$$\begin{cases} \{p_r^{(l)(d)(i)} - p_r^{(l)(d)(i-1)} = 0\}|_{i=e-1}^{e} & D-Rl, where\ e>2. \\ \{|p_r^{(l)(d)(i)} - p_r^{(l)(d)(i-1)}| < W_r^{(l)(d)}\}|_{i=e-1}^{e} & C-RL, where\ e>2. \end{cases} \tag{12}$$

We formally define this deep switching condition or termination condition as $deep\ switching\ number = 3$, and discuss parameter sensitivity in Section 5.5.

To acquire better training results, we use white-box methodology to test the results synchronously during the training process, and verify whether the convergence value is optimal for this round before starting a new round of RL for the same relation. If the value is negative, the filtering threshold with better performance in the historical version will be reviewed in the new round of optimization as the basis for the new round of action range. For this backtracking mechanism, we will conduct a sensitivity experiment in Section 5.5.

In this subsection, we overcome the second and third challenges mentioned in Sec. 2.2. Specifically, we use the similarity measure in the previous subsection to perform Top-p sampling for neighbors of each relation. Based on reinforcement learning, we use the agent to interact with the environment to make different relations to obtain different threshold combinations. This adaptive method is free from the help of data annotation. Furthermore, in order to meet the accuracy requirements of different relations while ensuring accuracy, we propose a recursive framework for optimization.

3.3 Relation-aware Weighted Neighbor Aggregator

Based on the selection of similar neighbors for each relation, the next step is to aggregate all these neighbor information among relations, for a comprehensive embedding. Previous methods employ attention mechanisms [27, 33, 59, 98] or weighting parameters [60] to learn the relation weights during the aggregating procedure. To reduce the computational cost whilst retaining the relation importance information, we directly use the optimal filtering threshold $p_r^{(l)}$ learned by the RSRL process as the inter-relation aggregation weights. Formally, for central node v, under relation r at the l-th layer, the intra-relation neighbor aggregation can be defined as follows:

$$\mathbf{h}_{v,r}^{(l)} = ReLU(AGG_r^{(l)}(\{\oplus\, \mathbf{h}_{v'}^{(l-1)} : v' \in \mathcal{N}_r^l(v)\})), \tag{13}$$

where $\oplus$ denotes the embedding bitwise summation operation for mean aggregator $AGG_r^{(l)}$, and $\mathcal{N}_r^l(v)$ refers to the set of top $p_r^{(l)}$ nodes obtained by Eq. (10) under relation r at the l-th layer. The purpose of the intra-relation neighbor aggregation for the central node v is to aggregate all neighborhood information under the relation r at the previous layer into the embedding vector $h_{v,r}^{(l)}$.

To follow up, we define inter-relation aggregation as below:

$$\mathbf{h}_v^{(l)} = ReLU(\mathbf{h}_v^{(l-1)} \oplus AGG^{(l)}(\{\oplus\, (p_r^{(l)} \cdot \mathbf{h}_{v,r}^{(l)})\}|_{r=1}^{R})), \tag{14}$$

where $h_{v,r}^{(l)}$ indicates the intra-relation neighbor embedding at the l-th layer and $AGG^{(l)}$ can be any type of aggregator. Here we directly use the $p_r^{(l)}$ optimized by the RSRL framework as the aggregation weight, and conduct experiments on other types of aggregation methods in Section 5.1.1 and Section 5.2.1.

In this subsection, in order to better deal with improper nodes to respond to Challenge 1 in Sec. 2.2, we divide the aggregation process into inter-relation and intra-relation, and use filtered neighbor nodes and filter thresholds of different relations to strengthen the influence of benign nodes during aggregation.

3.4 Put Them Together

We denote the final embedding of node v as $\mathbf{z}_v = \mathbf{h}_v^{(L)}$, which is the output of RioGNN at the last layer. We also define the loss function of GNN in node classification task as the cross-entropy loss function:

$$\mathcal{L}_{GNN} = \sum_{v \in \mathcal{V}} -log(y_v \cdot \sigma(MLP^{(l)}(\mathbf{z}_v))). \tag{15}$$

Algorithm 1: RemGNN: Reinforced Neighborhood Selection Guided Multiple Relation GNN.

Require : An undirected multi-relational graph with node features and labels: $\mathcal{G} = \{\mathcal{V}, \mathbf{X}, \{\mathcal{E}_r\}|_{r=1}^{R}, Y\}$;
Number of layers, batches, epochs: L, B, E; Parameterized similarity measures: $\{S^{(l)}(\cdot,\cdot)\}|_{l=1}^{L}$;
Filtering thresholds: $P = \{p_1^{(l)}, \ldots, p_R^{(l)}\}|_{l=1}^{L}$; Intra-R aggregators: $\{\text{AGG}^{(l)}\}|_{r=1}^{R}, \forall l \in \{1, \ldots, L\}$;
Inter-R aggregators: $\{\text{AGG}_r^{(l)}\}, \forall l \in \{1, \ldots, L\}$;
RL Module: $\{f_r^{(l)(d)}|_{d=1}^{D_r^l}\}, \forall r \in \{1, \ldots, R\}, \forall l \in \{1, \ldots, L\}$.

Ensure : Vector representations $\mathbf{z}_v, \forall v \in \mathcal{V}_{train}$.

// Initialization
$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$; $p_r^{(0)} = 0.5, d_r^{(0)} = 0, \mathcal{E}_r^{(0)} = \mathcal{E}, \forall r \in \{1, \ldots, R\}$
$\{p_r^{(l)(0)} \in [0, 1]\}, \forall r \in \{1, \ldots, R\}, \forall l \in \{1, \ldots, L\}$
// Training process of the proposed RioGNN
for $e = 1, \cdots, E$ **do**
  for $b = 1, \cdots, B$ **do**
    for $l = 1, \cdots, L$ **do**
      $\mathcal{L}_{\text{Simi}}^{(l)} \leftarrow$ Eq. (4) // Cross entropy loss of label-aware similarity measure
      for $r = 1, \cdots, R$ **do**
        $S^{(l)}(v, v') \leftarrow$ Eq. (3) , $\forall (v, v') \in \mathcal{E}_r^{(l-1)}$;
        // Edge set under relation r at the l-th layer
        $\mathcal{E}_r^{(l)} \leftarrow$ *top-p* sampling (Section 3.2.1);
        $\mathbf{h}_{v,r}^{(l)} \leftarrow$ Eq. (13) $\forall v \in \mathcal{V}_b$ // Intra-relation aggregator
      $\mathbf{h}_v^{(l)} \leftarrow$ Eq. (14) $\forall v \in \mathcal{V}_b$; // Inter-relation aggregator
    $\mathbf{z}_v \leftarrow \mathbf{h}_v^{(L)}, \forall v \in \mathcal{V}_b$; // Batch node embeddings
    $\mathcal{L}_{\text{GNN}} \leftarrow$ Eq. (15); // Cross-entropy loss function of GNN
    $\mathcal{L}_{\text{RioGNN}} \leftarrow$ Eq. (16); // Final loss function of RioGNN
  // RSRL Module: Markov Decision Process for filtering threshold
  for $l = 1, \cdots, L$ **do**
    for $r = 1, \cdots, R$ **do**
      if $d_r^{(l)} < Dr^{(l)}$ **then**
        // Judgement of the termination condition
        if Eq. (12) is False **then**
          $s_r^{(l)(d)(e)}, g_r^{(l)(d)(e-1)} \leftarrow$ Eq. (8) and Eq. (9) // Calculate state and reward
          $f_r^{(l)(d)(e)} \leftarrow$ Eq. (10); // Update RL iterative function
          // the recursive process of optimal proportion of neighbor nodes
          $p_r^{(l)(d)(e)} \leftarrow$ Eq. (11), $p_r^{(l)(d)} \in \{p_r^{(l)(d-1)} - \frac{W_r^{(l)(d)}}{2}, p_r^{(l)(d-1)} + \frac{W_r^{(l)(d)}}{2}\}$
        else
          $d_r^{(l)} = d_r^{(l)} + 1$;// Update the depth of Reinforcement Learning Tree

Together with the loss function of node classification and the loss function of the similarity measure in Eq. 4, we define the final loss function of RioGNN as follow:

$$\mathcal{L}_{\text{RioGNN}} = \mathcal{L}_{GNN} + \lambda_l \sum_{l=1}^{L} \mathcal{L}_{Simi}^{(l)} + \lambda_* ||\Theta||_2, \tag{16}$$

where λ_l and λ_* are the weight parameters, and $||\Theta||_2$ is the $L2$-norm of all model parameters.

Finally, based on the aforementioned study, Algorithm 1 outlines the training process of the proposed RioGNN which takes any given input multiple relation graph built upon a real-world practical problem. We employ the mini-batch training technique [28] to cope with excessively large real-world graphs. We initialize the parameters of the label-aware similarity module, relation-aware neighbor select module, and relation-aware neighbor aggregator module, before training the RioGNN model at each epoch. For each batch of nodes, we first compute the neighbor similarities using Eq. (4) and then leverage the *top-p* sampling to filter the neighbors. Thereafter, we compute the intra-relation embeddings (by using Eq. (13)) and inter-relation embeddings (by using Eq. (14)), and define the loss functions (by using Eq. (16)) for the current batch. In the RSRL module, we allocate H RL modules in sequence according to the maximum depth for each layer of each relation. In each RL module, each epoch will observe the environment state via Eq. (8) and get the reward via Eq. (9). Then the algorithm updates the iterative function through Eq. (10) and predict the filtering threshold via Eq. (11) of the current epoch through the updated iterative function. When an RL module reaches the convergence condition through Eq. (12) without targeting the maximum depth, we will recursively proceed to the next depth RL until all RL modules complete.

Time Complexity of RioGNN. The overall time complexity of Algorithm 1 is $O(|\mathcal{E}| \cdot \max(\{\alpha \log_\alpha k_r\}|_{r=1}^{R}))$, where $|\mathcal{E}|$ is the number of edges, α is the weight parameter of depth first and breadth first, and k_r is the maximum number of neighbors contained in the node in relation r. Here, $O(|\mathcal{E}|)$ is the time complexity in one epoch. Specifically, the similarity measure (Line 10 in Algorithm 1) and aggregation (Line 13-15 in Algorithm 1) take a total of $O(|\mathcal{E}|d + |\mathcal{V}|(\bar{k}d + d)) = O(|\mathcal{E}|)$, where $\bar{k}$ is the average node degree and $|\mathcal{V}|$ is the number of nodes. The RSRL module (Line 21-29 in Algorithm 1) takes $O(|\mathcal{E}|d) = O(|\mathcal{E}|)$. Each cross-entropy loss function (Line 8 and Line 16 in Algorithm 1) takes $O(|V|)$. In addition, the number of epochs is affected by the action space of reinforcement learning, and the number of epochs required to achieve convergence is related to the action space with the greatest demand among several relations $\max(\{\alpha \log_\alpha k_r\}|_{r=1}^{R})$.

4 EXPERIMENTAL SETUP

In the following two sections, we conduct experiments to evaluate and test RioGNN. The experimental setup mainly revolves around the following six questions:

- **Q1:** How do we build multi-relational graphs in different scenarios (Section 4.2)?
- **Q2:** The effectiveness, efficiency and explainability of RioGNN in fraud detection tasks (Section 5.1).
- **Q3:** The effectiveness, efficiency and explainability of RioGNN in disease detection tasks (Section 5.2).
- **Q4:** How do different task requirements match the universal RSRL framework (Section 5.3)?
- **Q5:** How does our model perform in clustering tasks and inductive learning (Section 5.2.1, Section 5.1.1 and Section 5.4)?
- **Q6:** Discussion of hyper-parameter sensitivity and effects on the model (Section 5.5).

4.1 Experimental Settings

We implement RioGNN with Pytorch. All experiments are running on Python 3.7.1, and a NVIDIA V100 NVLINK GPU with 32GB RAM. The operating system is Ubuntu 20.04.2. To improve the training efficiency and avoid overfitting, we employ mini-batch training and under-sampling techniques to train RioGNN and other baselines.

Specifically, under each mini-batch, we randomly sample the same number of negative instances as the number of positive instances.

4.2 Datasets and Graph Construction

We build different multi-relational graphs for experiments in *two* task scenarios and *three* datasets. Table 2 lists various statistical information of different dataset nodes and relationships. In addition to the number of nodes and the proportion of noisy nodes (Fraud%, Diabetes%) in different scenarios, we also give the number of edges with different relations. For each relationship in each dataset, we calculate the feature similarity of adjacent nodes based on the Euclidean distance (range 0 to 1) of the feature vector of adjacent nodes, and normalize the average feature similarity. The last column of Table 2 shows the average label similarity of each relationship, which is calculated based on whether two connected nodes have the same label.

Table 2. Dataset and graph statistics.

Dataset	#Nodes (Fraud% / Diabetes%)	Relation	#Edges	Avg. Feature Similarity	Avg. Label Similarity
Yelp	45,954 (14.5%)	R-U-R	49,315	0.83	0.90
		R-T-R	573,616	0.79	0.05
		R-S-R	3,402,743	0.77	0.05
		ALL	3,846,979	0.77	0.07
Amazon	11,944 (9.5%)	U-P-U	175,608	0.61	0.19
		U-S-U	3,566,479	0.64	0.04
		U-V-U	1,036,737	0.71	0.03
		ALL	4,398,392	0.65	0.05
MIMIC-III	28,522 (49.9%)	V-A-V	152,901,492	0.62	0.54
		V-D-V	19,183,922	0.63	0.54
		V-P-V	149,757,030	0.63	0.54
		V-M-V	15,794,101	0.65	0.51
		ALL	337,636,545	0.63	0.53

4.2.1 Fraud Detection Task. We perform binary classification tasks, spam review detection and fraudulent user detection on the Yelp dataset and Amazon dataset, respectively.

Yelp: is collected from the internal dataset published by Yelp.com, the largest business reviewing site in the United States. We use a subset of the YelpChi dataset collected and used by [70]. This subset contains 45,954 user reviews of hotels and restaurants in the Chicago area. The reviews have been filtered (spam) and recommended (legal) by Yelp. In addition to containing information about the relations between users and products, the dataset also contains various metadata, including the text content of the review, timestamp, and star rating. We use 32 manual features including *the ranking order of all product reviews*, *the absolute rating deviation from the product average*, *whether it is the only review of the user*, *the percentage of all uppercase words*, *the percentage of uppercase letters*, *the length of the review*, *the ratio of first-person pronouns*, *the ratio of exclamatory sentences*, and *subjective Word ratio*, *ratio of objective words*, etc. used in [84] as the original node features of the Yelp dataset. For specific relations, since previous research [70, 84] has shown opinion fraudsters (i.e., spammers) are connected in terms of users, products, review text, and time, we use reviews as nodes in the graph and design three relations:

- R-U-R: it connects reviews posted by the same user.
- R-T-R: it connects two reviews under the same product posted in the same month.
- R-S-R: it connects reviews under the same product with the same star rating (1-5 stars).

Amazon: is a subset of Amazon's product dataset [67]. The Amazon dataset contains more than 34,000 consumer reviews, from which we extracted 11,949 product reviews under the musical instrument category. In addition, similar to [116], we mark users with useful voting rates greater than 80% as benign entities, and users with useful voting rates less than 20% as fraudulent entities. In terms of node feature selection, we use 25 manual features including *the number of rated products*, *the length of the user name*, *the number and ratio of each rating level given by the user*, *the ratio of positive and negative reviews*, *the user's rating*, *the total number of useful and useless votes obtained by the user*, *the ratio of useful votes and useless votes*, and *Average value*, *median of useful and useless votes*, *minimum and maximum number of useful and useless votes*, *number of days between the user's first and last rating*, *same date indicator*, *comment text sentiment*, etc. used in [116] as the original node function of the Amazon data set. For specific relations, we design three kinds of relations for the multi-relational graph, as shown below:

- U-P-U: it connects users reviewing at least one same product.
- U-S-V: it connects users having at least one same star rating within one week.
- U-V-U: it connects users with the top 5% mutual review text similarities (measured by TF-IDF) among all users.

4.2.2 Diagnosis of diabetes mellitus task. We perform the binary classification task of diagnosis of diabetes mellitus on the processed MIMIC-III dataset resources.

MIMIC-III: is a publicly available dataset [26, 45] consisting of health records of 46,520 intensive care unit (ICU) patients over 11 years. We have extracted a total of 28,522 patient visits, and each record contains information such as age, diagnosis, microbiology, procedures, corpus, etc. We use a subset of the MIMIC-III dataset collected and used by [62], and construct our multi-relational graph for our task based on that. For each patient and visit, there is a unique ID to track its corresponding information. According to the diagnostic codes, we mark the medical records as diabetic or non-diabetic. Ages are split into groups using threshold 15, 30 and 64 as suggested in [64]. Procedures and diagnoses are mapped into corresponding ICD-9-CM codes. Microbiology tests with culture-positive results are mapped into the names of organisms. We use the above four fields to form different relationships among visit nodes to construct a heterogeneous graph. Based on previous medical representation learning, we obtain the feature representation of each node based on the medical corpus obtained from the admission record. Due to the complexity of the medical knowledge field, we appropriately enlarge the selection range of the relationship to further test the filtering performance of our RSRL Framework for camouflaged neighbors. For specific relations, since the previous work [6] has described the concept of different fields in detail, we design four kinds of relations for the multi-relational graph based on it, as shown below:

- V-A-V: it connects visits in the same age category.
- V-D-V: it connects visits having the same diagnoses.
- V-P-V: it connects visits with at least one same procedure code.
- V-M-V: it connects visits with at least one same microbiology code.

4.2.3 Evaluation of Datasets. We measure different datasets from the four metrics of node distribution, edge distribution, and feature similarity and label similarity. It can be observed that in fraud detection tasks, Yelp and Amazon have only 14.5% and 9.5% of fraud nodes, while mimic has a more balanced ratio. In addition, the different relations of the three datasets all have different number levels and uneven edge distribution. In addition, in the Yelp and Amazon datasets, edges with different relations have balanced feature similarity, but existing edges have higher feature similarity and lower label similarity. For instance, the average feature similarity of R-T-R is 0.79, but the label similarity is only 0.05. This shows that the fraudsters successfully pretends to be in benign entities and needs a more effective way to identify it. In general, these characteristics challenge the model's ability to learn from data sets in different situations. Specifically, Yelp and Amazon focus on the ability

to challenge uneven data sets. Mimic focuses more on the ability of filtering high-density neighbor nodes. The relation matrix of MIMIC under each relation is denser, one order of magnitude higher than that of Yelp and Amazon.

4.3 Baselines and Variations

4.3.1 Baselines. To verify the effectiveness of RioGNN in mitigating mutual interference between similar tasks and the model, we compare it with traditional and latest GNN baselines under semi-supervised learning settings. For all baseline models, we use the open-source implementations.

The first three models are compared as traditional GNN baselines.

- **GCN** [49]: is a representative of the spectral graph convolution method, which sets up a simple and well-behaved hierarchical propagation rule for neural network models. This rule runs directly on the graph, and uses the first-order approximation of the Chebyshev polynomial to complete an efficient graph convolution architecture.
- **GAT** [96]: is a neural network architecture combined with the attention mechanism that runs on graph-structured data. It uses masked self-attentional layers to give importance to the edges between nodes, help the model learn structural information, and assign different weights to different nodes in the neighborhood without expensive calculations and pre-definitions.
- **Graph-SAGE** [32]: is a representative non-spectrogram method. For each node, this method provides a general inductive framework that samples and aggregates its local neighbors' features to generate the embedding instead of training a separate embedding. It improves the scalability and flexibility of GNNs.

The second ten baselines are the latest GNN models that handle multi-relational data or the datasets used in this article.

- **RGCN** [85]: is a relational GCN model that uses Gaussian distribution as the hidden layer node feature representation, and relies on the attention mechanism to automatically assign the weight of each neighbor to aggregate neighbor information.
- **GeniePath** [59]: is a scalable graph neural network model used to learn the adaptive receptive domain of neural networks defined on permutation invariant graph data. Through the breadth and depth exploration of an adaptive path layer, the model can sense the importance of neighboring nodes and extract and filter signals gathered from the neighborhood.
- **Player2Vec** [118]: is an AHIN representation learning model that maps the attribute heterogeneous information network (AHIN) to a multi-view network, encodes the correlation between users described by different design meta-paths, and uses the attention mechanism to fuse embeddings from each view to form the final node representation.
- **SemiGNN** [98]: is a semi-supervised attention graph neural network, in which a hierarchical attention mechanism is designed. Neighborhood information is integrated through node-level attention, and multi-view data is integrated through view-level attention, resulting in better accuracy and interpretability.
- **GAS** [53]: uses both a heterogeneous graph and a homogeneous graph to capture the local and global context of a comment, and is a meta-path based heterogeneous GCN model. In our scenario, meta-paths are enumerated from relations.
- **FdGars** [99]: constructs a single homogeneous graph based on multiple relations and employs GNNs to aggregate neighborhood information. Compared with our work, this model lacks neighborhood selection.
- **GraphConsis** [61]: is a model that combines context embedding with nodes, filters inconsistent neighbors and generates corresponding sampling probabilities. The embeddings of sampled nodes from each relation are fused using a relation attention mechanism.

Table 3. Comparison of main functions of different variants.

Models	**Multi-layer**	**RL Module**	**Action Space**	**Recursion**	**Inter-AGG**
RioGNN $_{2l}$	✓	AC	Discrete	✓	Threshold
BIO-GNN	×	BMAB	Discrete	✓	Threshold
ROO-GNN	×	AC	Discrete	×	Threshold
RIO-Att	×	AC	Discrete	✓	Attention
RIO-Weight	×	AC	Discrete	✓	Weight
RIO-Mean	×	AC	Discrete	✓	Mean
RioGNN	×	AC	Discrete	✓	Threshold

- **HAN** [101]: is a hierarchical attention network aggregating neighbor information via different meta-paths. Although the input data is heterogeneous, meta-paths are symmetrical in our scenario (i.e., the end nodes are of the same type), thus the model is regarded as a homogeneous model.
- **GCT** [13]: is a basic model for learning the implicit EHR structure using Transformer. Using statistical data to guide the structure learning process solves the problem that existing methods require complete docking structure information. Specifically, they use the attention mask and prior knowledge to guide self-attention to learn the hidden EHR structure, and they can learn the underlying structure of the EHR together even when the structural information is missing.
- **HSGNN** [62]: is a heterogeneous medical graph-based semi-supervised graph neural network, which combines both meta-path instance based similarity matrices and self-attention mechanism.

The third two baselines are the latest Reinforcement Learning guided GNN models. We respectively use the raw heterogeneous graph and the proposed multi-relation graph as the input of these models.

- **GraphNAS** [24]: enables automatic search of the suitable graph neural architecture via reinforcement learning. This model uses a recurrent network to generate variable-length strings that describe the architectures of graph neural networks, and then trains the recurrent network with reinforcement learning to maximize the expected accuracy of the generated architectures.
- **Policy-GNN** [51]: is a meta-policy framework that adaptively learns an aggregation policy to sample diverse iterations of aggregations for different nodes. To accelerate the learning process, we also use a buffer mechanism to enable batch training and parameter sharing mechanism to decrease the training cost.

The last baseline is from the preliminary version of this article.

- **CARE-GNN** [15]: A layer of label-perceived similarity measure is used to find information-rich neighboring nodes. Then the Bernoulli Multi-armed Bandit (BMAB) mechanism is used to explore the optimal number of neighbors for each relationship.

Among those baselines, GCN, GAT, GraphSAGE, and GeniePath runs on the homogeneous graph (i.e., Relation ALL in Table 2), where all relations are merged together. GraphNAS^H and Policy-GNN^H runs on the raw heterogeneous graph. Other models run on the multi-relational graph. On the multi-relational graph, they process information from different relations in their methods.

4.3.2 Variations. We have implemented many variants of the RioGNN model. The configurations of different variants are shown in Table 3. In detail, the setting of the model variants mainly revolves around the key mechanisms of the three modules: Label-aware Similarity Measure, Similarity-aware Neighbor Selector, and Relation-aware Neighbor Aggregator.

The first one given is a variation of the unexpanded multi-layer version of the Label-aware Similarity Measure section.

- **RioGNN $_{2l}$:** It uses the Actor-Critic (AC) algorithm with a discrete strategy to recursively select the filter thresholds of different relationships, and uses the filter thresholds as relation weights to aggregate neighbors between different relations. But the Label-aware Similarity Measure uses a 2-layer structure for neighbor selection.

The next two methods are variants according to the Similarity-aware Neighbor Selector module.

- **BIO-GNN:** This variant is a method with the single-layer similarity-aware neighbor selection and uses filtering thresholds for aggregation between relations. But the Bernoulli Multi-armed Bandit (BMAB) algorithm of discrete strategy is used to recursively select the filtering threshold of relations.
- **ROO-GNN:** This variant is a method with the single-layer similarity-aware neighbor selection and uses filtering thresholds for aggregation between relations. But the Actor-Critic algorithm of discrete strategy is used to directly select the filter threshold of relationships. This method can be regarded as a non-recursive (one-depth) version of RioGNN.

The difference between the following three variants is reflected in the aggregation method between different relations of Relation-aware Neighbor Aggregator.

- **RIO-Att:** This variant uses single-layer similarity perception for neighbor selection, and uses the Actor-Critic algorithm with a discrete strategy to recursively select the filter thresholds of different relations. But it chooses the method of Attention [96] when aggregating neighbors between different relations.
- **RIO-Weight:** This variant uses single-layer similarity perception for neighbor selection, and uses the Actor-Critic algorithm with a discrete strategy to recursively select the filter thresholds of different relations. But it chooses the method of Weight [60] when aggregating neighbors between different relations.
- **RIO-Mean:** This variant uses single-layer similarity perception for neighbor selection, and uses the Actor-Critic algorithm with a discrete strategy to recursively select the filter thresholds of different relations. But it chooses the method of Mean [32] when aggregating neighbors between different relations.

4.3.3 RL Variations. In order to better discuss the adaptability of the framework of this article to a variety of reinforcement learning algorithms, we conducted experiments on two action spaces (discrete and continuous) of different reinforcement learning algorithms.

The first three reinforcement learning models are based on discrete action spaces. That is, in the process of constructing the reinforcement learning forest, a discrete filtering threshold is used as the action type of reinforcement learning.

- **AC [50]:** Actor-Critic (AC) method combines the advantages of the value-based method and policy-based method. The value-based method is used to train the Q function to improve the sample utilization efficiency. The policy-based method is used to train the strategy, which is suitable for discrete and continuous action spaces. This kind of method can be regarded as an extension of the value-based method in the continuous action space, or as an improvement of the policy-based method to reduce the sampling variance.
- **DQN [69]:** Deep Q-Learning (DQN) is a temporal-difference, value-based and off-policy reinforcement learning method. DQN approximately solves the dimensional disaster problem of Q-Learning method in the face of high-dimensional state and action through the function approximation. In addition, the traditional Q-Learning method uses samples with time series for single-step update, and the Q value is updated by the sample continuity. DQN uses random data for gradient descent due to trial and error to collect a large number of samples, which can break the correlation between data.
- **PPO [86]:** Proximal Policy Optimization (PPO) restricts the update step size on the basis of Policy Gradient (PG) to prevent policy collapse and make the algorithm rise more steadily.

The next four reinforcement learning models are based on continuous action spaces. That is, in the process of constructing the reinforcement learning forest, a continuous filtering threshold is used as the action type of reinforcement learning.

- **AC [50]:** We use the AC method to conduct variation experiments in both discrete action space and continuous action space.
- **DDPG [55]:** Deep Deterministic Policy Gradient (DDPG) is an off-policy algorithm for continuous control developed by DeepMind, which is more sample efficient than PPO. DDPG trains a deterministic policy, that is, only one optimal action is considered in each state.
- **SAC [36]:** Soft Actor-Critic (SAC) is an off-policy algorithm developed for Maximum Entropy Reinforcement learning. Compared with DDPG, Soft Actor-Critic uses stochastic policy, which has certain advantages over deterministic policies. Soft Actor-Critic has achieved outstanding results in the public benchmark and can be directly applied to real robots.
- **TD3 [23]:** Twin Delayed Deep Deterministic policy gradient (TD3) is a temporal-difference, policy based and policy gradient reinforcement learning method. TD3 is an optimized version of DDPG. It uses two sets of networks to estimate the Q value, and the relatively smaller one is used as the update target.

4.4 Model Training

We use unified embedding size (64), batch size (1024 for Yelp, 256 for Amazon and MIMIC-III), learning rate (0.01), the similarity loss weight ($\lambda_1 = 2$), L2 regularization weight ($\lambda_2 = 0.001$) for all the models. Except for the comparative experiments specifically explained, other experiments that are not explained all use a 40% training ratio, under-sampling ratio $1:1$, deep switching number 3, weight parameter of depth first and breadth first α as 10, and a single-layer similarity perception structure with backtracking. For the reinforcement learning model, we use gamma (0.95), learning rate (0.001) as unified parameters and buffer capacity (5), batch size (1) as parameters for DDPG, DQN, SAC and TD3. We conduct the sensitivity study for deep switching number, backtracking setting, under-sampling ratio in Section 5.5. In addition, we also discuss weight parameter of depth first and breadth first in Section 5.3, and training ratio in Section 5.1 and Section 5.2.

4.5 Evaluation Metrics

We utilize ROC-AUC (AUC) [95] and Recall to evaluate the overall performance of all classifiers. AUC is computed based on the relative ranking of prediction probabilities of all instances, which could eliminate the influence of imbalanced classes. The Recall is defined as:

$$Recall = \frac{TP}{TP + FN}, \tag{17}$$

where TP is True Positive, FN is False Negative. The AUC is defined as:

$$AUC = \frac{1}{2}\sum_{i=1}^{m-1}(x_{i+1} - x_i)(y_i + y_{i+1}), \tag{18}$$

where y is True Posotive Rate ($TPR = \frac{TP}{TP+FN}$), x is False Positive Rate ($FPR = \frac{FP}{FP+TN}$). And FP is False Positive, TN is True Negative.

In the clustering task, we use Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) as the performance indicators. The ARI is defined as:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}, \tag{19}$$

where each n_{ij} represents the number of nodes located in class$_i$ and cluster$_j$ at the same time, a_i is the number of the nodes in class$_i$ and b_j is the number of the nodes in cluster$_i$. The NMI is defined as:

$$NMI = \frac{I(\omega; C)}{[H(\omega) + H(C)]/2}, \tag{20}$$

where I is mutual information, H is entropy.

In inductive learning, in order to better measure the effectiveness, we add the F1 indicator to measure after the two indicators of AUC and Recall. F1 is defined as:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recal}, \tag{21}$$

where $Precision = \frac{TP}{TP+FP}$, TP is True Positive, FP is False Positive.

5 RESULT AND DISCUSSION

5.1 Overall Evaluation of Fraud Detection Task

5.1.1 Accuracy Analysis. In this section, we conduct experiments to evaluate the accuracy of the fraud detection task on Yelp and Amazon datasets. We report the best test results of RioGNN, baselines and variants in five hundred epochs. It can be observed from the results that RioGNN performs better than other baselines and variants under most training ratios or indicators. This indicates the feasibility of RioGNN in fraud detection scenarios.

Single-relations vs. Multi-relations. Table 4 shows the results of baseline experiments built on different types of graphs for the fraud detection task. To solve the diversity and heterogeneity of complex networks in actual fine-grained applications, we consider introducing a neural network with a multi-relational graph structure instead of a single relation structure. However, from the results of some of the baselines in the table, although GCN, GAT, GraphSAGE and GeniePath models run on a single relation graph, they are better than RGCN, Player2Vec and SemiGNN in terms of accuracy of the Yelp dataset, which are run on the multi-relational graphs. The observation above shows that the previous multi-relational GNNs are not suitable for constructing multi-relational graphs in the fraud detection task. Similar phenomena manifest in the Amazon dataset. Besides, GraphConsis, FdGars, CARE-GNN and RioGNN significantly outperform other models by 8.60%-32.78% over Yelp and Amazon datasets. This is because these four models sample the neighbors according to node features before aggregating them, indicating that the impurity neighbors will interfere with the aggregation process and the fraud detection task has a strong demand for neighbor sampling optimization. CARE-GNN and RioGNN have improved the AUC of 3.51%-21.65%, compared with the performance of GraphConsis and FdGars. This is because RioGNN can better use internal relations to solve downstream application problems through parameterized similarity measures and adaptive sampling thresholds, which shows that automated sampling has a significant improvement effect on fraud detection tasks. The more remarkable result is that the proposed RioGNN model improves the accuracy of 5.90% and 10.41% compared with CARE-GNN. It verifies the advantage of combining the label-aware neighbor similarity measure and the Recursive and Scalable Reinforcement Learning framework, which can effectively break through the limitations of the CARE-GNN state observation range and manually specified strategies. Meanwhile, RioGNN has a promising effectiveness on fraud detection tasks.

Heterogeneous vs. Multi-relation. In order to further analyze the accuracy of the multi-relational graph, we conduct heterogeneous graph experiments and multi-relational graph experiments on the latest GNN model guided by reinforcement learning. From the results of the heterogeneous graph model GraphNAS^H, Policy-GNN^H and the multi-relationship graph model GraphNAS and Policy-GNN in Table 4, it can be found that the multi-relational graph compared with the heterogeneous graph brings an AUC improvement of 0.33%-1.71% in the Yelp and Amazon datasets. This confirms that in other similar models, the multi-relational graph we construct

Models	Yelp								Amazon							
	AUC				Recall				AUC				Recall			
	5%	10%	20%	40%	5%	10%	20%	40%	5%	10%	20%	40%	5%	10%	20%	40%
GCN	54.98	50.94	53.15	52.47	53.12	51.10	53.87	50.81	74.44	75.25	75.13	74.34	65.54	67.81	66.15	67.45
GAT	56.23	55.45	57.69	56.24	54.68	52.34	53.20	54.52	73.89	74.55	72.10	72.16	63.22	65.84	67.13	65.51
GraphSAGE	53.82	54.20	56.12	54.00	54.25	52.23	52.69	52.86	70.71	73.97	73.97	75.27	69.09	69.36	70.30	70.16
RGCN	50.21	55.12	55.05	53.38	50.38	51.75	50.92	50.43	75.12	74.13	75.58	74.68	64.23	67.22	65.08	67.68
GeniePath	56.33	56.29	57.32	55.91	52.33	54.35	54.84	50.94	71.56	72.23	71.89	72.65	65.56	66.63	65.08	65.41
Player2Vec	51.03	50.15	51.56	53.65	50.00	50.00	50.00	50.00	76.86	75.73	74.55	56.94	50.00	50.00	50.00	50.00
SemiGNN	53.73	51.68	51.55	51.58	52.28	52.57	52.16	50.59	70.25	76.21	73.98	70.35	63.29	63.32	61.28	62.89
GraphConsis	61.58	62.07	62.31	62.07	62.60	62.08	62.35	62.08	85.46	85.29	85.50	85.50	85.49	85.38	85.59	85.53
GAS	54.43	52.58	52.51	52.60	53.40	53.26	53.37	51.61	71.40	77.49	74.51	71.03	64.31	64.57	62.08	63.74
FdGars	61.77	62.15	62.81	62.66	62.83	62.16	62.73	62.40	85.58	85.41	85.88	85.81	85.83	85.73	85.84	85.93
GraphNAS^H	52.93	54.69	56.73	54.46	52.40	54.15	55.69	56.16	71.01	72.48	73.52	76.05	69.17	69.48	70.35	70.16
GraphNAS	53.26	55.31	57.15	55.59	53.69	55.47	56.04	57.00	72.41	73.04	73.58	76.25	70.36	70.53	71.73	71.88
Policy-GNN^H	54.04	55.73	59.30	60.60	53.08	55.35	58.75	59.99	72.20	73.30	74.11	77.20	70.10	71.20	73.08	74.44
Policy-GNN	55.75	56.29	60.01	61.52	54.15	56.16	58.95	60.33	73.69	74.06	75.29	78.85	71.34	72.46	74.55	76.70
CARE-GNN	71.26	73.31	74.45	75.70	67.53	67.77	68.60	71.92	89.54	89.44	89.45	89.73	88.34	88.29	88.27	88.48
RioGNN	**81.97**	**83.72**	**82.31**	**83.54**	**75.33**	**75.78**	**75.51**	**76.19**	**95.44**	**95.41**	**95.63**	**96.19**	**90.17**	**89.48**	**89.51**	**89.82**

Table 4. Fraud Detection results (%) compared to the baselines.

Table 5. Fraud Detection classification results (%) compared to RioGNN variants.

Models	Yelp		Amazon	
	AUC	**Recall**	**AUC**	**Recall**
RioGNN $_{2l}$	76.01	63.15	91.28	72.46
BIO-GNN	78.67	71.21	95.47	88.35
ROO-GNN	**83.59**	**75.56**	95.58	89.22
RIO-Att	78.65	71.69	93.97	83.78
RIO-Weight	80.40	72.83	**96.25**	**89.61**
RIO-Mean	77.84	71.43	94.57	89.47
RioGNN	83.54	75.55	96.19	88.66

still has obvious advantages. In addition, we found that GraphNAS and Policy-GNN, which are also based on reinforcement learning guidance and use the multi-relational graph, have no significant advantages in AUC and Recall. This is because GraphNAS and Policy-GNN do not adaptively sample different relations, which causes them to be limited by the complexity of the relationship between Yelp and Amazon datasets. And for Policy-GNN, since the one-hop neighbor information of Yelp and Amazon is already rich enough, more multi-hop strategies cannot bring significant benefits.

Training Percentage. To measure the impact of the training ratio on the classification accuracy, we use four different ratios of 5% to 40% for experiments. It can be seen from Table 4 that most of the baseline performance changes are not necessarily related to the increase in training percentage. It indicates that the semi-supervised learning approach leveraging a small number of supervised signals is enough to train a good model. Moreover, in the four different training ratios of the Yelp dataset, the AUC fluctuation range of RioGNN is only within 1.57% compared to the 4.44% of CARE-GNN. In Amazon, both models have good stability. This is because the Amazon node features provide enough information to distinguish fraudsters, which is of higher quality than the Yelp dataset. It also verifies from another result that RioGNN has better stability and adaptability under complicated environments.

RioGNN Variants in Classification. To measure the positive impact of the newly added mechanism on the classification accuracy of fraud detection tasks, we compare several variants of RioGNN under the discrete strategy. The experiment sets the training data ratio to 40%, and the other settings are the same as Section 4.1. We show the experimental results of spam review classification in the Yelp dataset and suspicious user classification in the Amazon dataset as shown in Table 5. From the results, the performance of all variants is better than the baseline model. Next, we will discuss the effects of different variants from three aspects.

Firstly, in the two datasets, except for the RioGNN $_{2l}$ variant, all other variants only use the Label-aware Similarity Measure with a one-layer structure. From the results in Table 5, the RioGNN $_{2l}$ variant is lower than all other variants, but it performs better than all baselines. It can be found that in the fraud detection task, the increase in the number of layers does not bring about a significant increase in classification accuracy. This is limited by the dataset size of Yelp and Amazon, and the importance of information in multi-hop neighbors is low. We will continue to explore more multi-layer effects in Section 5.1.3.

Secondly, for the similarity-aware neighbor selector part, we observe the comparison results of the variant BIO-GNN without adaptive strategy optimization reinforcement learning algorithm and the single-depth structure variant ROO-GNN without the recursive framework for the RSRL framework. The second part of Table 5 gives evidence of partial optimization. The results of the BIO-GNN variant on the RioGNN model show that the automatic strategy optimization in Yelp and Amazon effectively improves the classification accuracy of 4.65% and 0.89%, and the BIO-GNN variant is also far better than most baseline models. This shows that the Markov Decision Process has a positive effect on searching for the filtering threshold of the aggregation process. Moreover, the

Table 6. Fraud detection clustering results (%) compared to RioGNN variants.

Dataset	Metric	RioGNN $_{2l}$	BIO-GNN	ROO-GNN	RIO-Att	RIO-Weight	RIO-Mean	RioGNN
Yelp	**NMI**	3.18	9.36	**12.39**	9.80	12.05	8.39	12.22
	ARI	6.12	11.84	**16.61**	11.88	15.88	8.80	16.45
Amazon	**NMI**	58.87	59.83	57.81	55.76	58.76	58.72	**61.26**
	ARI	76.53	77.38	76.09	76.54	76.73	76.51	**78.40**

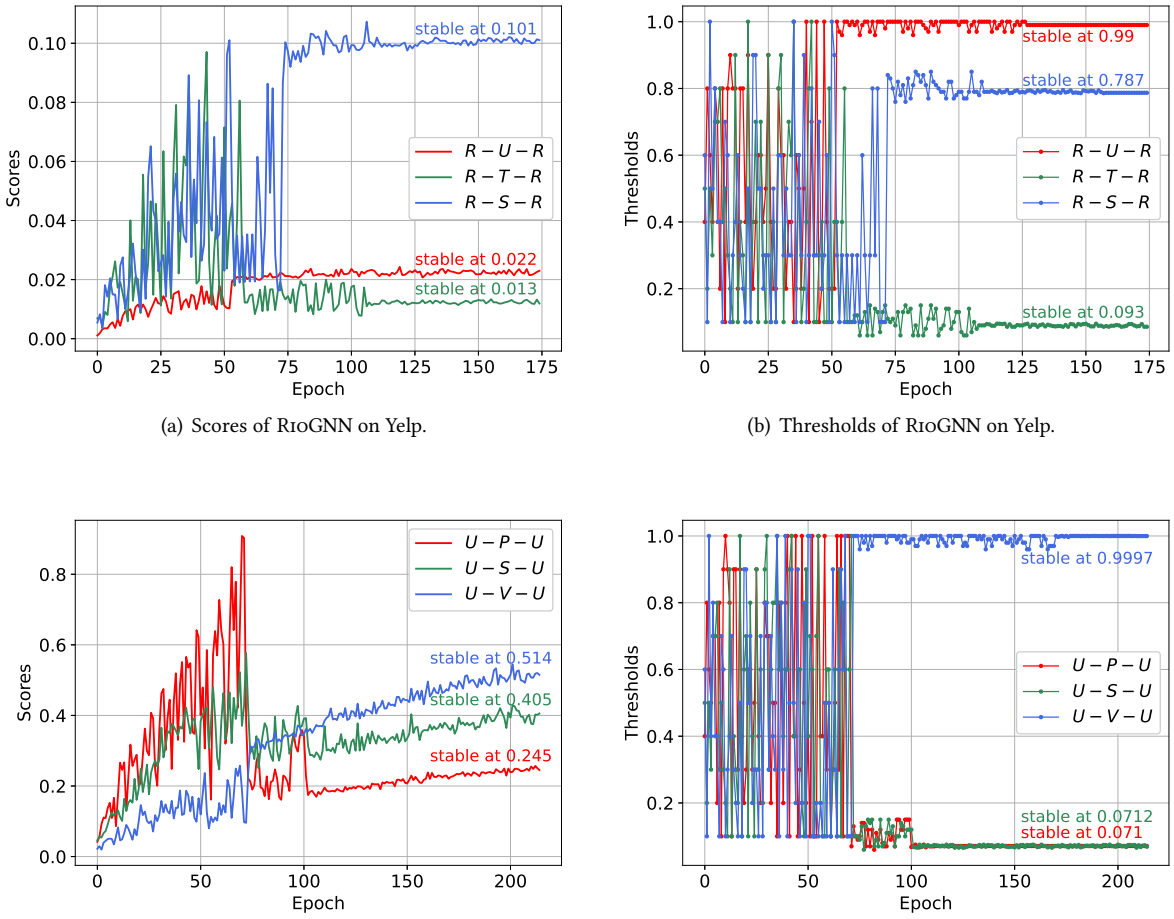
reinforcement learning algorithm with dynamic iterative function and full action space learning process breaks the limitation of fixed strategy and observation range and obtains a better threshold selection effect, which also confirms the conjecture in Section 3.2.2. Besides, combining Figure 8 with Table 5, the accuracy of the ROO-GNN variant has little change compared with RioGNN, that is, the multi-depth structure of RioGNN can converge much quicker than the single-depth variant ROO-GNN whilst maintaining a higher accuracy rate. This also implies the stability of the recursive framework in terms of classification accuracy.

Finally, from the results of the variation of the aggregation methods between the different relations in the third part of the Table 5, RioGNN has apparent advantages over the other three on the Yelp dataset, and both RIO-Weight and RioGNN on the Amazon dataset have good results. It confirms that RioGNN does not need to train additional attention weights, and using the filtering threshold as an inter-relation aggregation weight can improve the performance of GNN and reduce the complexity of the model. Therefore, it can get the best performance compared with other variants. For the three variants, RIO-Weight has better results than the other two, but RioGNN can maintain better accuracy in the dataset of different quality and structure and has a certain degree of adaptability.

RioGNN Variants in Clustering. In order to explore the effectiveness of RioGNN in clustering tasks, we conduct clustering experiments on RioGNN and its variant models. The experiment set a fixed training rate of 40%. We cluster the node representations learned by RioGNN through K-Means. The results are shown in Table 6. We respectively count the best values of NMI and ARI indicators within 500 epochs. It can be seen from the results that compared with RIO-GNN$_{2l}$ and BIO-GNN, RioGNN's NMI and ARI indicators in the Yelp dataset increase at least 9.04% and 10.33% respectively. Similarly, the Amazon dataset has risen by at least 2.39% and 1.87%. This phenomenon is the same as the classification result, and is affected by the limitation of the size of the dataset, the choice of the action space, and the dynamic iterative function. In addition, the NMI and ARI of ROO-GNN in the Yelp dataset achieved the best results, while the RioGNN in the Amazon dataset exceeded the NMI and ARI of ROO-GNN by 3.45% and 2.31%. This is because the Yelp dataset is smaller than Amazon, so the optimization space of the recursive framework is also smaller. It shows that the recursive framework has better advantages in dense datasets. What is more noteworthy is that RioGNN in the clustering experiment exceeds the effect of all GNN aggregation variants. NMI and ARI exceed 0.17%-3.83% and 0.57%-7.65% in the Yelp dataset, and exceed 2.5%-5.5% and 1.67%-1.89% in the Amazon dataset. This shows that directly using the filtering threshold as the weight of the aggregation has obvious advantages in clustering tasks.

5.1.2 Explainable RSRL Training Process. This section focuses on the RSRL framework and discusses the explainability of the reinforcement learning process in detail. We show the change process of the RioGNN's filtering threshold and similarity score of the different relations before convergence during the training process on the Yelp and Amazon datasets. For a better comparison, we also conduct a similar analysis on ROO-GNN and BIO-GNN variants.

Filter Thresholds. In Table 2, we observe that the average feature similarity for most relations in Yelp and Amazon are very high. However, there is a relation such as R-T-R, which has a low label similarity of 0.05, which indicates that fraudsters successfully disguised themselves. For this reason, we propose to filter the lower-ranked neighbors in the Top-p sampling through the filtering threshold. It can be seen from Figure 5(b) and Figure 5(c)



(a) Scores of RioGNN on Yelp.

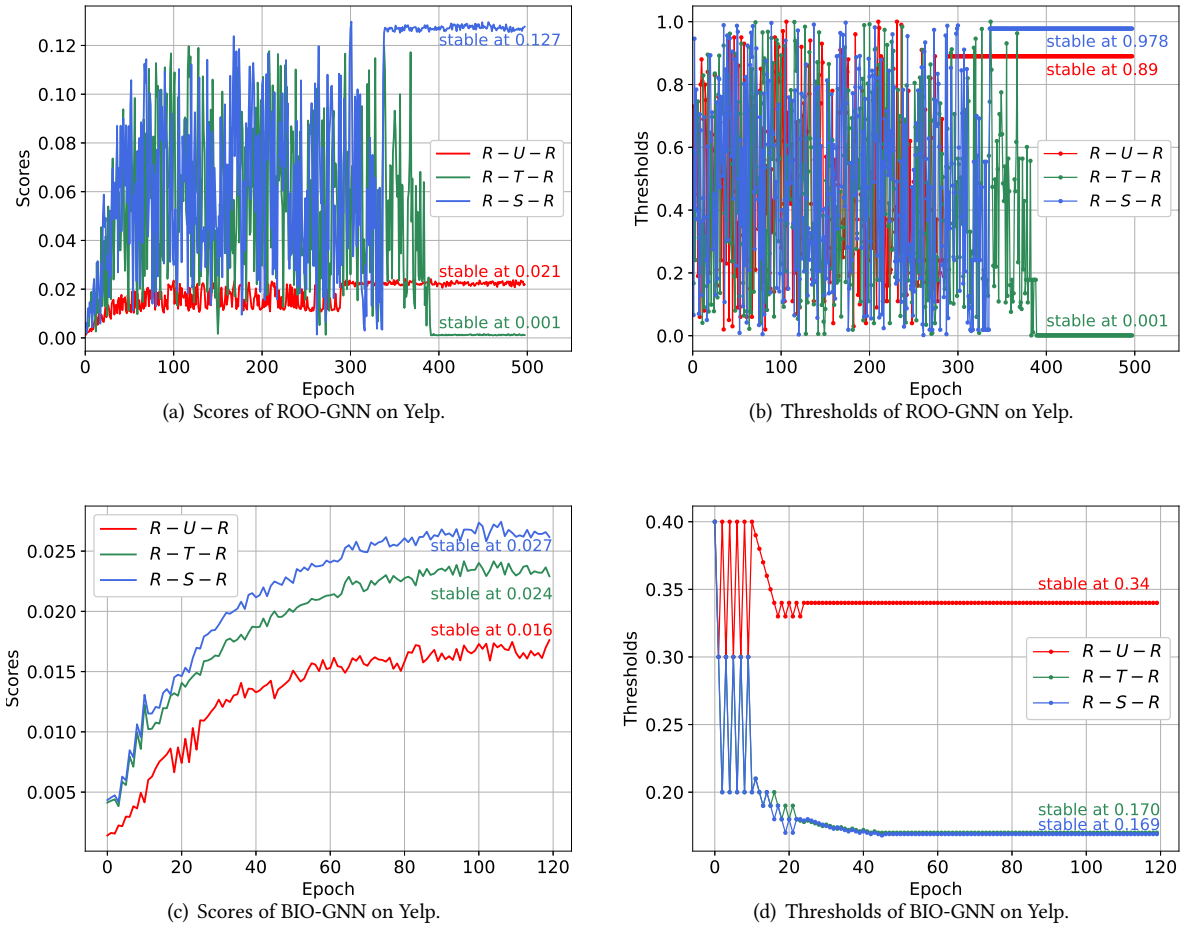
(b) Thresholds of RioGNN on Yelp.

(c) Scores of RioGNN on Amazon.

(d) Thresholds of RioGNN on Amazon.

Fig. 5. The training scores and thresholds of RioGNN on Yelp and Amazon.

that the filtering thresholds of the three relations of the Yelp dataset are stable at $[0.99, 0.093, 0.787]$, and the Amazon dataset converges to $[0.071, 0.0712, 0.9997]$. It shows that the filtering thresholds for different relations eventually converge to different values. The reason is that the label similarity and feature similarity of different relations are different in the same dataset. This result also can be verified from Table 2. For instance, the label similarity difference between R-U-R and R-T-R is 0.85, but the feature similarity difference is only 0.04. The proposed framework builds a reinforcement learning tree for each relation, uses the similarity between the relations as a reward, and independently finds the appropriate filtering threshold. Due to the mutual influence between relations, a set of Nash equilibrium filtering thresholds will eventually be obtained, which is a sampling scheme that can best eliminate the interference of fraudsters [16]. In the Nash equilibrium, it is impossible for all agents to obtain greater rewards only by changing their own strategies. In addition, in Figure 5(b) and Figure 6(b),

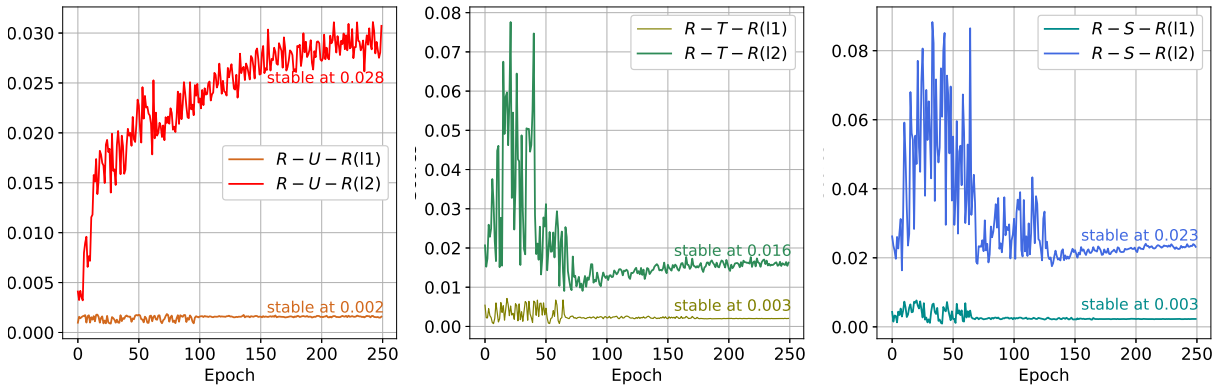


(a) Scores of ROO-GNN on Yelp.
(b) Thresholds of ROO-GNN on Yelp.
(c) Scores of BIO-GNN on Yelp.
(d) Thresholds of BIO-GNN on Yelp.

Fig. 6. The training scores and thresholds of RioGNN variants on Yelp.

Figure 6(d), we also observe that different models are used under the same dataset, and they converge and combine with different filtering thresholds $[0.99, 0.093, 0.787]$, $[0.89, 0.001, 0.978]$, $[0.34, 0.17, 0.169]$, respectively. Since the result of reinforcement learning is obtained by the connection strategy of all agents, there may be many different filter threshold combinations at each time the game between relations reaches the Nash equilibrium [40, 82].

Figure 5(a) and Figure 5(c) show the changes in rewards obtained during the filtering threshold learning process for different relations. We observe that in the Yelp dataset, the relations R-S-R and R-U-R achieve better reward growth in the interest competition of the three relations. In contrast, the relation R-T-R eventually stabilizes at a relatively low reward. This is consistent with the actual scenario. Comments with the same star rating in the same product are important considerations for dividing spam comments, and comments by the same user usually have the same tendency. However, reviews published in the same month have a lower impact factor. The U-P-U and U-S-U of the similar Amazon dataset are more meaningful than the relation U-V-U. This means that users



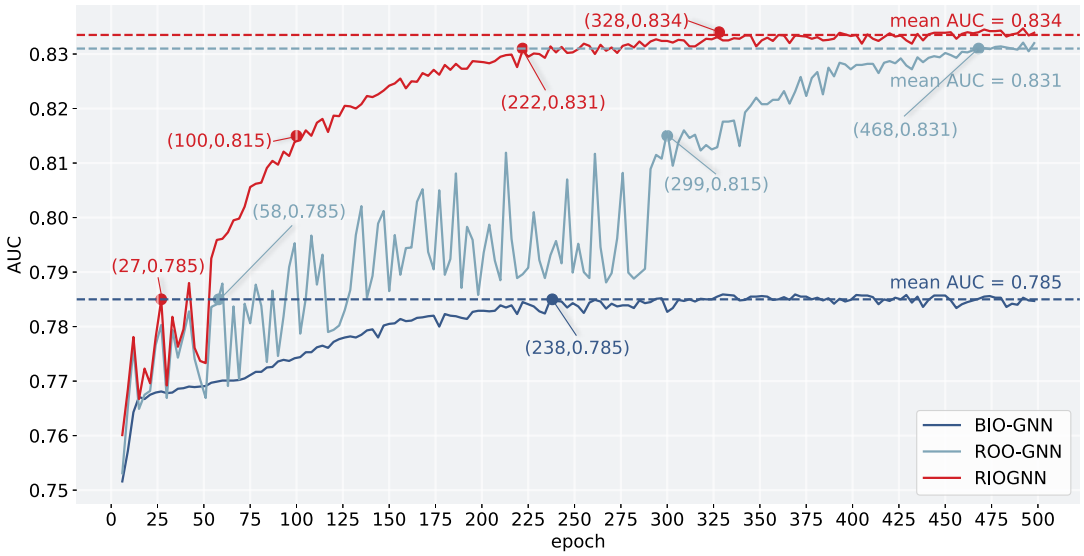
(a) R-U-R. (b) R-T-R. (c) R-S-R.

Fig. 7. Scores of Multi-Layer RioGNN on Yelp.

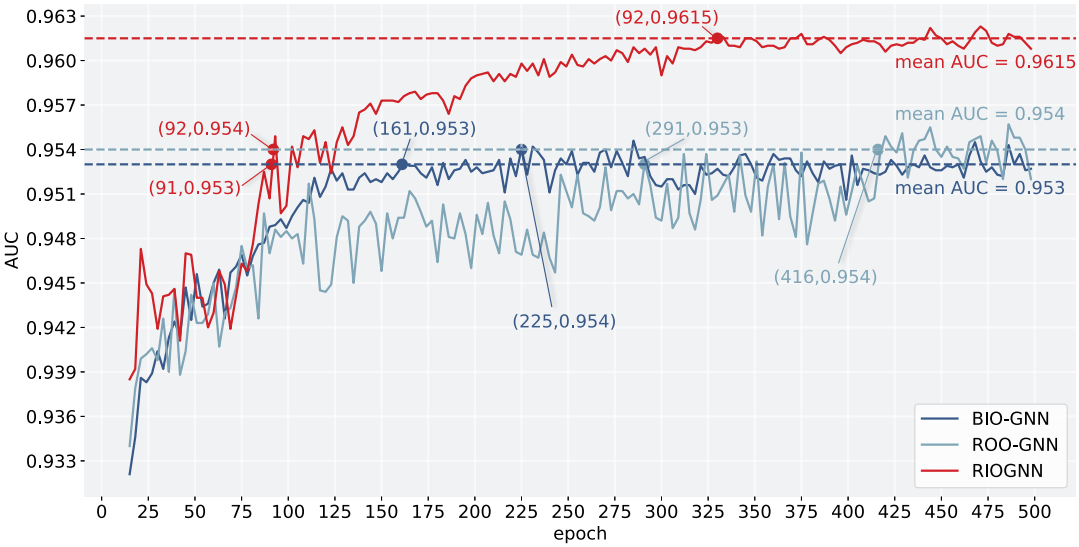
who post similar content are more closely connected, and are considered an important observation factor when making user judgments.

Recursive Framework. In the right column of Figure 6, we show the changes in the filtering thresholds of the two variants of RSRL, ROO-GNN and BIO-GNN on the Yelp dataset. Comparing RioGNN in Figure 5(b) with ROO-GNN in Figure 6(b), it can be seen that RioGNN with the recursive framework of Figure 5(b) performs a more accurate filtering threshold search for each depth. For example, the relation R-U-R is first explored in the interval $[0, 1]$ with an accuracy of 0.1 to obtain a convergence of 0.9. Then it searches for the convergence between $[0.85, 0.95]$ to obtain the highest accuracy 0.01 of the relation R-U-R, and get the final convergence 0.99. The difference from RioGNN is the ROO-GNN model in Figure 6(b), where only one deep reinforcement learning is performed. For example, R-U-R directly explores the $[0, 1]$ interval with the highest accuracy of 0.01 and obtains the final convergence value of 0.89. In addition to the difference mentioned above, we can also see whether there is a recursive reward change shown in Figure 5(a) and Figure 6(a). For RioGNN with recursion, all relations converge fully at 110 epochs, while ROO-GNN without recursion converges at 390 epochs. Furthermore, the rewards obtained by the two methods are basically the same. This solves the challenge we pose in Section 2.2, and explains that reducing the number of actions for each reinforcement learning can effectively accelerate the convergence of the model. The experimental results also show that the addition of recursion does not bring about a significant loss of accuracy, which is significant.

Action Space and Iterative Function. In Figure 5(b) and Figure 6(d), we present the BIO-GNN variant and RioGNN with dynamic iterative function and full action space learning process. It can be seen that, similar to RioGNN, BIO-GNN also recursively converges to 0.4 with an accuracy of 0.1 for the R-U-R relation and then converges to 0.33 with an accuracy of 0.01. But the difference from RioGNN is that this method has an average similarity score of 0.045 for the three relations, which is lower than RioGNN's 0.021. This means that the maximum filtering effect has not been achieved. In other respects, compared with the Figure 5(a) and Figure 6(a) that have obtained better performance, the R-T-R relation of Figure 6(c) has obtained a higher reward ratio in the competition. That is, the behaviors published in the same month are considered by BIO-GNN to be more important, which is contrary to the reality. So this explains the reason for the lower performance of BIO-GNN.



(a) AUC of Rio-GNN, BIO-GNN and ROO-GNN on Yelp.



(b) AUC of RioGNN and RioGNN $_{No\ Recursion}$ on Amazon.

Fig. 8. The impact of recursive framework on computational efficiency.

Meanwhile, these results demonstrates that dynamic learning can be observed globally, thereby obtaining more effect, which proves the conjecture in Section 3.2.2.

5.1.3 Effectiveness and Efficiency Evaluation. Next, we introduce the effectiveness and efficiency of multi-layer similarity perception modules and recursive neighbor selectors in fraud detection tasks on the two datasets. We use RioGNN with a two-layer perception structure to train for 500 epochs on the Yelp dataset, and select the first 250 epochs to show the changes in the scores of each layer in Figure 7. For the recursive framework, we analyze the AUC change trend of ROO-GNN, BIO-GNN variants and RioGNN on two datasets within 500 epochs (Figure 8). The dotted line in the figure indicates the average AUC of each variant after reaching stability. The specially marked points in the figure show the epoch numbers for different models to reach a certain AUC.

Multi-layer Analysis. We provide a multi-layer label-aware similarity neighbor measurement scheme to deal with data collections with more complex structures in the future. For the datasets in this article, the increase of multiple layers is limited by the datasets (one-hop neighbor information is enough for sampling), and the AUC gain brought by it is limited. However, we notice in Figure 7 that as the number of layers increases, Layer 2 has a significant improvement in the similarity score of each relation compared to Layer 1. Among them, the similarity score of the relation R-U-R is 14.00 in the second level compared with the first level. In addition, the R-T-R and R-S-R are improved in 5.33 and 7.66. This is because we take the embedding of the neighbors of the previous layer as the input of the second layer, embedding more hops of neighbor relations. Rich neighbor information makes the performance of the similarity module better. This also shows that multi-layer joining can be a new deployment scheme in some datasets with insufficient embedded information of one-hop neighbors.

Multi-depth Analysis. In order to effectively speed up the optimization speed of the filtering threshold of each relation while ensuring the accuracy rate, we propose a recursive reinforcement learning learning framework. It can be seen from Figure 8 that the RioGNN model has obvious advantages over ROO-GNN and BIO-GNN in both datasets. First of all, we observe the final convergence AUC size. In the Yelp dataset, RioGNN and BIO-GNN models are about 4.75% higher than ROO-GNN. In the Amazon dataset, RioGNN is better than the other two models by about 0.8%. And BIO-GNN finally converged AUC is generally low. In addition, in terms of computational efficiency, compared with the ROO-GNN model without recursion, RioGNN maintains a stable and rapid increase in AUC in both datasets. However, ROO-GNN has greater fluctuations. In the first 50 epochs of Yelp and the first 75 epochs of Amazon, there is no significant difference between them. In the Yelp dataset, the speedup of RioGNN compared to ROO-GNN is 2.14 when AUC reaches 78.5%, 2.99 when AUC reaches 81.5, and 2.10 when AUC reaches 83.1%. In Amazon, the speedup ratio is 3.19 when the AUC reaches 95.30%, and the speedup ratio is 4.52 when AUC reaches 95.4%. Compared with the BIO-GNN with limited action space and fixed strategy, Yelp and Amazon are also observed 8.81 and 1.71 times time savings at 78.5% and 95.3% AUC. This shows that the proposed recursive framework can achieve good efficiency while maintaining accuracy. And generally, the higher the AUC demand, the better the efficiency. The broader and flexible action space and the iterative function that automatically updates have more significant advantages in terms of efficiency and accuracy. Finally, we find that different optimization structures have different impact factors for different datasets. Due to the smaller scale and lower accuracy requirements of the Yelp dataset, whether it has a recursive structure has little effect on the final convergence AUC. But in Amazon, which has a larger scale and higher precision requirements, placing all actions at one depth causes a loss of accuracy. This also confirms the conjecture in Section 3.2.2 about the loss of accuracy caused by the excessively large action space, and confirms the advantages of the recursive structure in large-scale datasets.

5.2 Overall Evaluation of Diagnosis of Diabetes Mellitus Task

5.2.1 Accuracy Analysis. In this section, we conduct experiments to evaluate the accuracy of diagnosis of diabetes mellitus on the MIMIC-III dataset. As presented in Table 7, we report the best test results of RioGNN and various baselines and variants in seven hundred epochs. It can be observed from the results that RioGNN performs better than other baselines and variants under most training ratios and indicators.

Table 7. Diabetes Detection results (%) compared to the baselines.

Models	MIMIC-III							
	AUC				Recall			
	5%	10%	20%	40%	5%	10%	20%	40%
GCN	65.37	65.39	64.93	65.13	60.85	61.24	60.31	60.99
GAT	63.67	63.09	63.11	64.26	62.13	62.75	63.12	63.45
GraphSAGE	65.91	66.28	65.82	65.34	63.80	63.88	63.82	63.99
GCT	62.97	63.45	64.33	65.14	60.08	61.23	61.39	62.25
HSGNN	63.27	65.68	65.03	67.87	61.24	63.87	64.08	65.39
HAN	62.79	63.26	64.94	65.13	61.15	61.27	62.38	63.02
GraphNAS^H	64.03	65.28	65.08	65.59	62.18	63.94	64.05	65.14
GraphNAS	65.76	67.40	67.79	68.05	64.28	66.88	67.31	67.93
Policy-GNN^H	66.30	67.19	67.70	67.98	63.28	66.05	67.12	68.18
Policy-GNN	67.45	68.55	69.01	69.59	64.73	67.11	68.32	69.02
CARE-GNN	77.64	80.22	80.81	81.26	69.17	71.58	72.08	72.68
RioGNN	**79.23**	**80.92**	**81.23**	**82.56**	**71.23**	**72.64**	**72.93**	**74.01**

Single-relation vs. Multi-relation. In order to further prove the effectiveness of RioGNN in processing the multi-relational graph, we perform a baseline comparison on the more challenging task of diagnosis of diabetes mellitus. Table 7 shows the comparative results of RioGNN and the three types of baselines. Compared with the first type of baseline running on a single-relational graph: GCN, GAT and GraphSAGE, the model RioGNN is significantly better than them on the MIMIC-III dataset by 13.32%-18.30%. This result fully confirms that the fine-grained division of multi-relational graph and hierarchical aggregation based on different relations are very conducive to the completion of the node classification task. This point is completely consistent with the experimental conclusions on the fraud detection task. Furthermore, in order to show the performance of RioGNN when dealing with the multi-relational graph, we carry out the second type of baseline comparison experiment. Although GCT, HSGNN and HAN all run on heterogeneous graphs, and propose different ideas for processing heterogeneous relations, their accuracy rates are at least 8.77% lower than RioGNN. Due to the high density of neighbor nodes under each relation, the inability to effectively filter interfering nodes also brings difficulties to the final diagnosis task. Compared with RioGNN, it is obvious that they fail to filter out interfering neighbor nodes and produce a sufficiently strong positive effect on the final diagnosis. Interestingly, comparing the two types of baselines, we find that although the second type of baselines divides the heterogeneous relations to a certain extent, they are in most cases even worse than the first type of baselines running on a single-relational graph. The above results show that when dealing with the multi-relational graph, how to effectively select the appropriate neighbor nodes is particularly important, while RioGNN achieves this well through parameterized similarity measures and adaptive sampling thresholds. When it comes to the third type of baseline, CARE-GNN, we find that its accuracy is significantly higher than the first two types of baselines by 11.73%-17.70%, which means that fine-grained and multi-relational division of heterogeneous graphs is very necessary. Although CARE-GNN realizes automatic filtering and sampling of neighbor nodes to a certain extent, its diagnostic effect is lower than RioGNN due to the inability to adaptively select the best filtering threshold under each relation. The above shows that the proposed RSRL framework finds the optimal filtering threshold under each relation in the recursive process successfully, thus it performs outstandingly in downstream tasks.

Heterogeneous vs. Multi-relation. In synchronization with the fraud detection task, we also analyze the performance of GraphNAS and Policy-GNN models in heterogeneous graph and multi-relational graph in the

Table 8. Diabetes diagnosis classification results (%) compared to RioGNN variants.

Models	MIMIC-III	
	AUC	**Recall**
RioGNN $_{2l}$	81.06	72.28
BIO-GNN	81.29	72.75
ROO-GNN	81.01	72.34
RIO-Att	80.96	72.16
RIO-Weight	81.04	72.58
RIO-Mean	80.31	77.42
RioGNN	**81.36**	**72.84**

disease detection task. From Table 7, the same phenomenon as the fraud detection task can be obtained, that is, the introduction of the multiple relational graph on the Mimic dataset also has certain advantages compared with the heterogeneous graph. However, unlike Yelp and Amazon, GraphNAS and Policy-GNN are generally better than other baselines in terms of accuracy. We believe that the cause is the balanced features and labels of the Mimic dataset with different relations, and the higher aggregation requirements of the large-scale dataset for neighbor information.

Training Percentage. In order to measure the impact of the training ratio on the classification accuracy in the diagnosis of diabetes mellitus task, we still set four different training ratios ranging from 5% to 40%. It can be seen from Table 7 that the classification accuracy of RioGNN shows a steady upward trend as the training ratio increases. This shows that the training process of RioGNN has a very positive effect on the final classification accuracy. It can be seen intuitively, RioGNN has successfully achieved a good performance improvement in the process of reinforcement learning recursion by supervised signals, which is also in line with expectations. However, the accuracy of some baselines changes unrelated to the training percentage, which means that their training methods have great limitations in this task and fail to improve the performance of their models by increasing the supervised signal learning process. It also verifies that RioGNN has better explainability and can continuously improve the accuracy of diagnosis through learning more node features. Consistent with its performance under the previous fraud detection task, RioGNN still maintains strong stability and explainability and its accuracy rate surpasses the others at each training ratio.

RioGNN Variants in Classification. In order to further verify the impact of the proposed new mechanism on different tasks, we compare different performances of several variants of RioGNN under the discrete strategy in the context of diabetes diagnosis. We show the experimental results of diagnosis of diabetes mellitus on the MIMIC-III dataset in Table 8. First of all, we compare three variants based on different aggregation methods on the MIMIC-III dataset. The results show that RioGNN is better than the other three variants, while RIO-Weight has better results than the other two, which is consistent with the results on Yelp dataset. We find that the choice of aggregation method is usually based on a specific dataset. As for different dataset, different relations have different ways of influencing the results, which lead to different aggregation methods. However, whether it is for Yelp, Amazon or MIMIC-III, the results show that RioGNN is superior to all aggregation variants, while RIO-Weight is second only. This point has a certain degree of universality. Next, we also focus on the performance of the RioGNN variant with two-layer structure, RioGNN $_{2l}$. Consistent with the previous results on Yelp and Amazon, the two-layer architecture doesn't bring very good performance improvements to the model on MIMIC-III. Compared with RioGNN, it can be found that the increase in the number of layers does not improve the final accuracy, indicating that the use of a single-layer structure based on the label-aware similarity measure for neighbor selection is optimal. However, in conjunction with Table 2, it is noted that MIMIC-III dataset is denser than Yelp and Amazon, thus the effect of RioGNN $_{2l}$ is very close to that of RioGNN. This shows that for

denser relations, the second-order neighbors found by RioGNN $_{2l}$ are more effective in the final result. To some extent, for too dense multi-relation graphs, second-order neighbors can be used as supplementary information for first-order neighbors. Finally, for the Label-aware Similarity Measure section, we observe the variant BIO-GNN without adaptive strategy optimization reinforcement learning algorithm and the single-depth structure variant ROO-GNN without the recursive framework for the RSRL framework again on MIMIC-III. It can be found from Table 8 and Table 7 that ROO-GNN performs significantly better than most baselines and variants, second only to RioGNN. This shows that Bernoulli Multi-armed Bandit (BMAB) algorithm of discrete strategy has strong adaptability in the process of recursively selecting the filtering threshold of relations. In contrast, the accuracy rate of ROO-GNN is 0.28% lower than BIO-GNN. Apart from that, Figure 10 shows that as the training epoch increases, ROO-GNN fluctuates greatly in the range of 79.48%-81.43%, and can never be stable in a fixed smaller interval as RioGNN or BIO-GNN. That is to say, the multi-depth structure of RioGNN brings better convergence speed and excellent stability than the single-depth variant ROO-GNN while maintaining a higher accuracy rate.

Table 9. Diabetes diagnosis clustering results (%) compared to RioGNN variants.

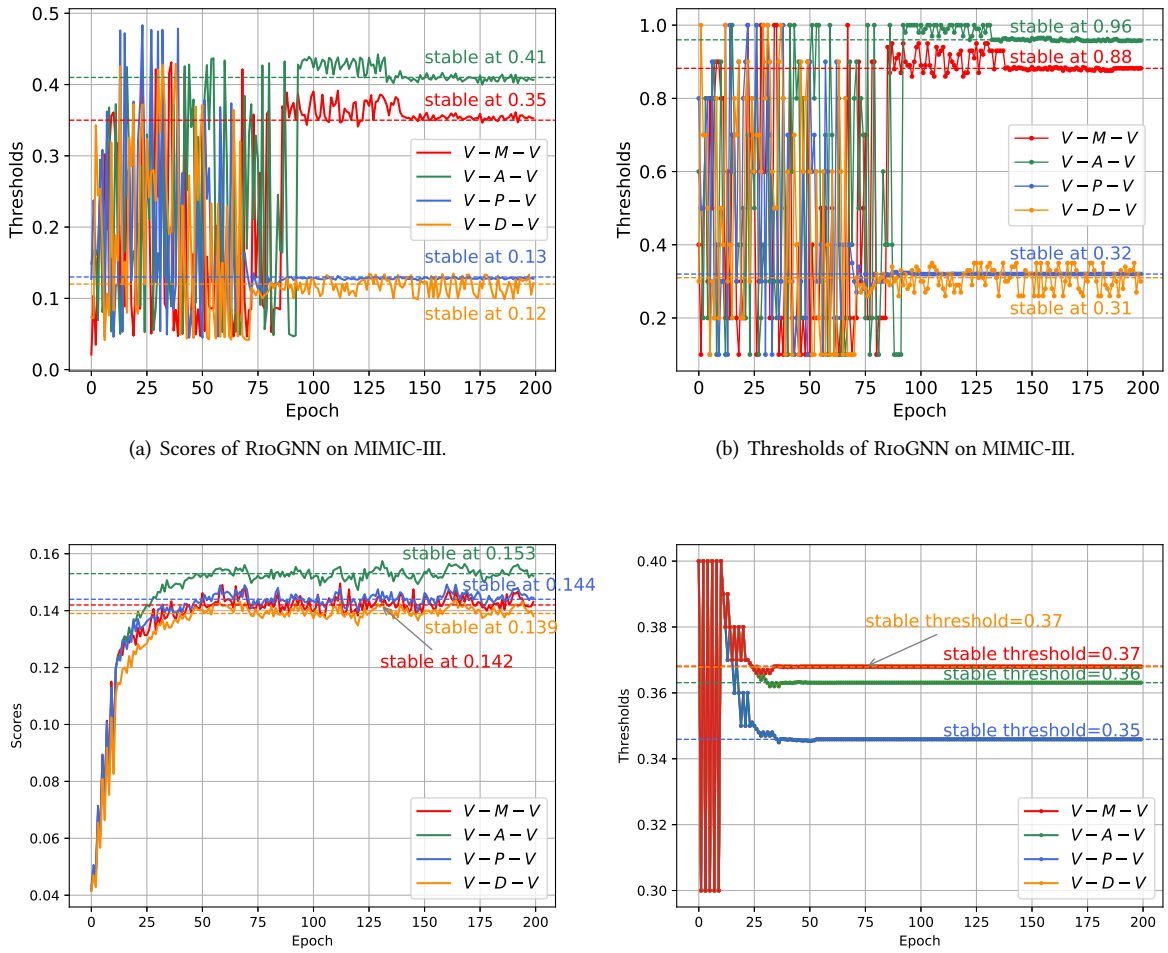
Dataset	Metric	RioGNN $_{2l}$	BIO-GNN	ROO-GNN	RIO-Att	RIO-Weight	RIO-Mean	RioGNN
MIMIC-III	**NMI**	19.01	19.81	19.13	17.17	**20.22**	19.86	20.10
	ARI	7.15	8.27	8.11	6.24	9.01	7.51	**10.03**

RioGNN Variants in Clustering. Similar to fraud detection, we also perform cluster analysis in the task of disease diagnosis. The best results of NMI and ARI within 700 epochs are recorded in Table 9. It can be seen that compared with BIO-GNN and ROO-GNN, RioGNN brings at least 0.29% and 1.76% increase in NMI and ARI respectively. This proves that the RSRL framework also brings accuracy optimization for dense datasets in the clustering task. In addition, RioGNN has better performance on ARI indicators for variants that use different methods for aggregation. For the NMI indicator, the RIO-Weight effect has been improved. We believe this is because the MIMIC-III dataset has a smaller difference in weight between the relationships compared with Yelp and Amazon. Weight can distinguish different classes better than RioGNN, which directly uses the filtering threshold as the aggregation parameter.

5.2.2 Explainable Reinforcement Learning Training Process. This section focuses on the process of reinforcement learning and explains the convergence process of proposed RioGNN on the MIMIC-III dataset. We also present a comparative analysis of different variants to further explain the applicability of RioGNN.

The Effectiveness and Necessity of the RSRL Framework. This part demonstrates the validity and necessity of the proposed framework by comparing RioGNN with variants and its preliminary version CARE-GNN. It can be seen from the Figure 10 that RioGNN performs better than CARE-GNN in almost every epoch, which proves that the RSRL framework has a positive effect on the final classification results on MIMIC-III. RioGNN can effectively filter suspected nodes by building a reinforcement learning tree for each relation and identify suspected nodes more accurately. Compared with RioGNN, the accuracy of ROO-GNN in Figure 9 fluctuates significantly, and it is difficult to converge to a stable range, which proves the necessity to establish a recursive process. Through the RSRL process, the classification accuracy can be maintained in a relatively stable range. The experimental results show that RioGNN with the recursive framework performs better through a more accurate filtering threshold search for each depth. While ROO-GNN without recursion not only fails to converge within a finite number of epochs but also causes a loss of accuracy.

Filter Thresholds. To further test the proposed model's filtering performance against suspected neighbors, we deliberately extract four denser relations when designing the MIMIC-III dataset. Each relation is at least an order of magnitude higher than the Yelp or Amazon dataset, which means that the MIMIC-III dataset is more challenging for RioGNN's filtering performance. It can be seen from Figure 9(b) and Figure 9(d) that the filtering



(a) Scores of RioGNN on MIMIC-III.

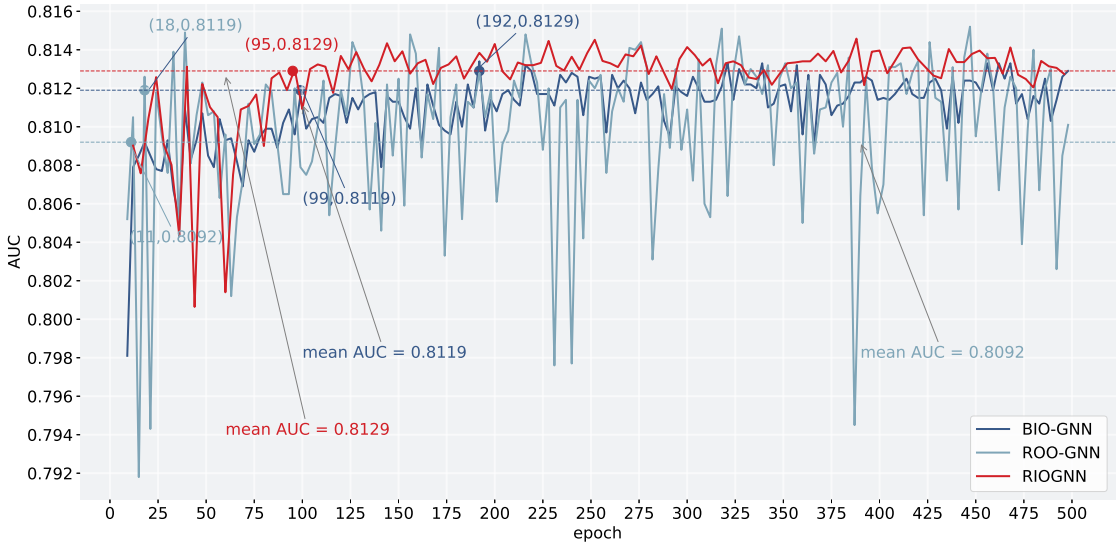
(b) Thresholds of RioGNN on MIMIC-III.

(c) Scores of BIO-GNN on MIMIC-III.

(d) Thresholds of BIO-GNN on MIMIC-III.

Fig. 9. The training scores and thresholds of RioGNN vs BIO-GNN on MIMIC-III.

thresholds of the four relations of RioGNN on the MIMIC-III dataset are stable at [0.88, 0.96, 0.32, 0.26], while BIO-GNN are stable at [0.35, 0.37, 0.36, 0.37]. Considering that the label similarity and feature similarity of different relations are different, the model's filtering strength for different relations is not the same. It is worth noting that there is a certain commonality between the relation filtering strength on RioGNN and BIO-GNN. Under RioGNN, the convergence thresholds of V-A-V and V-M-V are relatively similar, while V-P-V and V-D-V are relatively similar. Generally speaking, relations with high filtering thresholds can bring more positive guidance to the diagnosis result and are more explainable. From another perspective, a higher filtering threshold can prove the explainability and correctness of the selected relation, which is more conducive to us intuitively judging whether the choice of the relation is appropriate enough. It is clear from Figure 9 that these four relations of



(a) AUC of RioGNN,ROO-GNN and BIO-GNN on MIMIC-III.

Fig. 10. The effectiveness and necessity of the RSRL framework.

Table 10. Results (%) compared to different RL algorithms and strengthening strategies.

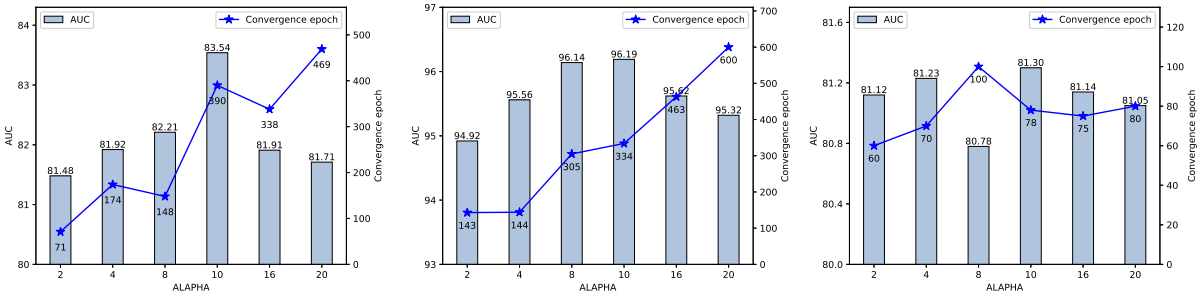
	Methods	Yelp	Amazon	MIMIC-III
Discrete	AC [50]	83.54	**96.19**	81.36
	DQN [69]	84.08	95.13	80.96
	PPO [86]	80.52	94.99	80.98
Continuous	AC [50]	81.31	94.72	80.98
	DDPG [55]	83.80	95.39	81.17
	SAC [31]	80.42	94.76	80.87
	TD3 [23]	**84.18**	95.11	**81.51**

RioGNN converge to a stable value within 100 epochs. This indicates that the algorithm can obtain a set of Nash equalization filter thresholds in a very limited epoch through RSRL framework, and thus has good stability and high efficiency.

5.3 Versatility Analysis of RSRL Framework

To better adapt to many task-driven scenarios, we implement a general RSRL reinforcement learning framework. In dealing with datasets of different sizes and types, different reinforcement learning algorithms and action space types can be flexibly matched. The depth and width of the reinforcement learning tree can be adaptively estimated for each relation.

Algorithms and Action Space for Different Task Scenarios. In Section 5.1 and Section 5.2, we analyze the function variants (Table 5 and Table 8) and the difference between reinforcement learning algorithms under RSRL framework and traditional reinforcement learning (Figure 8 and Figure 10) in terms of accuracy and



(a) Yelp. (b) Amazon. (c) MIMIC-III.

Fig. 11. Depth and Width for Different Task Scenarios.

efficiency through a classical reinforcement learning algorithm, Actor-Critic. Here, in order to better explore the versatility of RioGNN for different reinforcement learning algorithms and the applicability of different reinforcement learning algorithms for different tasks, in Table 10 we select the current mainstream reinforcement learning algorithms for experiments, and record the best AUC value within 500 epochs. In addition, for the RSRL framework, we proposed two different action spaces for the construction of reinforcement learning forests in Section 3.2.2 to adapt to different task requirements. Different from the discrete action space, the reinforcement learning framework with continuous action space has continuous precision (that is, the highest floating-point number of the processor) in every action selection of reinforcement learning. This difference makes it have a better exploration effect in large-scale dataset. From the experimental results, PD3 algorithm continuous action space and two sets of networks to update the Q value achieves the best results in both Yelp and MIMIC-III dataset. In the Amazon dataset, the best result is obtained from the more basic discrete action space AC. SAC and PPO are at a low level in the three datasets. In general, RioGNN is well-adapted to most reinforcement learning algorithms, and is a versatile framework for different types of dataset and task scenarios.

Depth and Width for Different Task Scenarios. In Section 3.2.2, we define a depth and width adaptive parameter α to adjust the size of the action space of each layer of the relation and the depth of the entire relation tree. In the previous experiment, we fixedly chose 10 as α. In this section, in order to discuss the impact of the depth and width adaptive parameter on the accuracy and efficiency of the RioGNN model, we compare and analyze the AUC and convergence epoch sizes of the three dataset under different settings. As shown in Figure 11, we set the six α values of 2, 4, 8, 10, 16, 20, and respectively record the maximum AUC and the corresponding epoch serial number obtained in 500 epochs. In the Yelp dataset, AUC achieves the maximum value when α is 10, which is at least 1.33% better than other parameters. But in this case, it takes longer to reach this value. Therefore, we suggest that Yelp can be adaptively chosen α to 8 or 10 for efficiency priority and accuracy priority. On the other hand, Amazon achieves better accuracy when α is 8 or 10, and achieves better efficiency when α is 2 or 4. The difference from the previous two is that the MIMIC-III dataset obtains a loss of accuracy and efficiency when α is 8. Judging from these situations, RioGNN can be adapted to a dataset of different scales and different accuracy and efficiency biases by adjusting α. This represents the versatility of the dataset size and task requirements.

Table 11. Inductive learning results (%) compared to RioGNN variants.

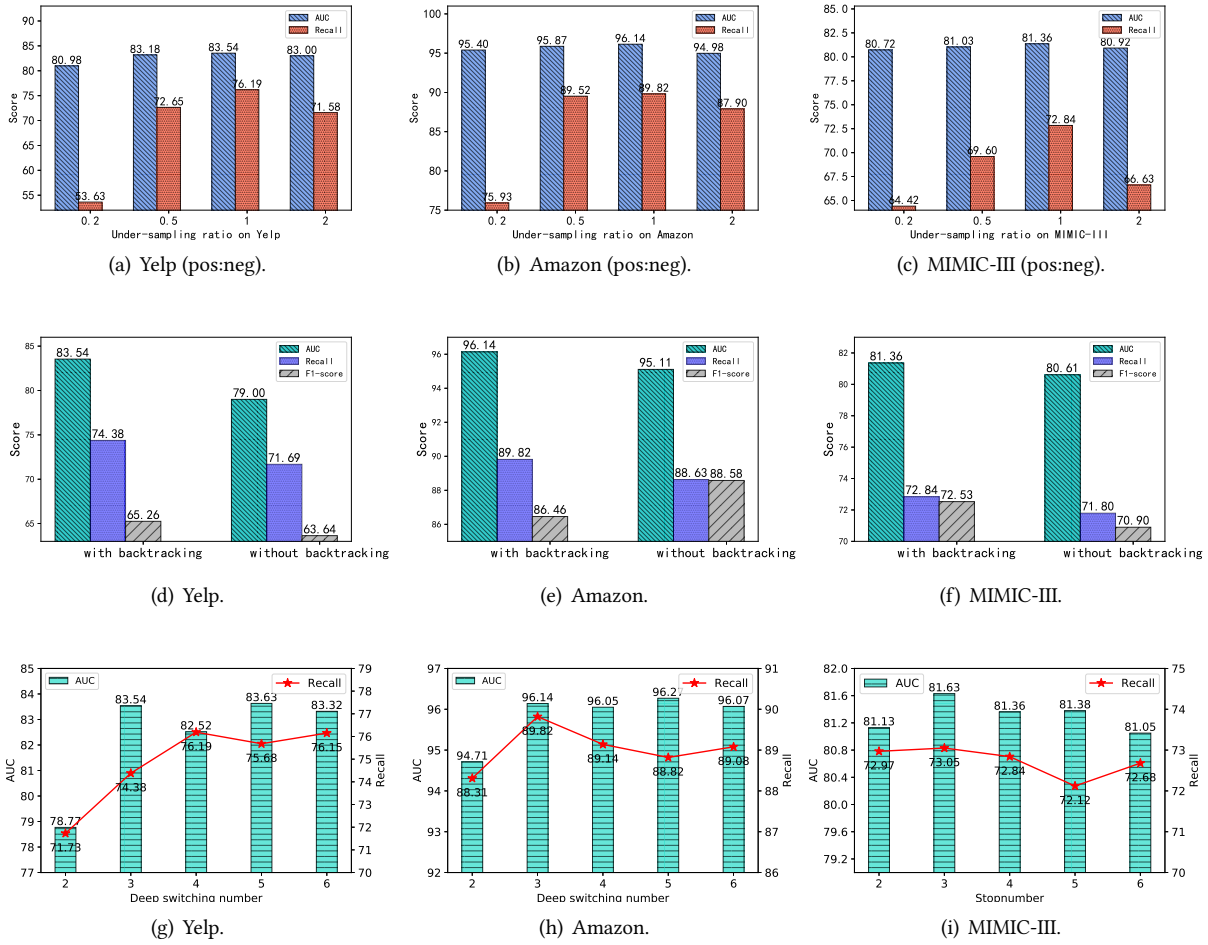
Models	Yelp			Amazon			MIMIC-III		
	AUC	**Recall**	**F1**	**AUC**	**Recall**	**F1**	**AUC**	**Recall**	**F1**
GAT	55.94	51.79	47.25	72.33	65.86	60.17	63.89	59.13	56.78
GraphSAGE	53.85	51.78	44.36	74.91	70.02	65.32	63.89	69.99	59.24
RioGNN $_{2l}$	79.45	71.86	63.58	92.01	83.65	86.24	79.01	69.77	69.64
BIO-GNN	79.49	71.86	63.58	**95.07**	88.19	86.51	81.21	**72.81**	**72.64**
ROO-GNN	82.15	74.23	**67.73**	94.79	87.43	**88.67**	81.01	72.39	72.23
RIO-Att	78.72	71.78	62.38	93.79	**88.71**	83.72	79.84	71.31	71.28
RIO-Weight	81.06	72.79	65.59	94.67	88.58	85.12	**81.25**	72.72	72.28
RIO-Mean	78.17	71.41	62.12	93.53	87.32	85.75	80.29	71.92	71.74
RioGNN	**82.38**	**75.08**	65.26	94.03	88.58	86.46	81.23	72.63	72.53

5.4 Inductive Learning Analysis

In this section, we perform inductive learning on RioGNN, some representative baselines and variant models of RioGNN. In the previous experiment, we use transductive learning, that is, the graph passed into the model contains the test nodes. In inductive learning, we only pass the adjacency matrix of the nodes that need to be trained into the model, and record the best AUC, Recall and F1 indicators of Yelp and Amazon within 500 epochs and MIMIC-III within 700 epochs. From the results in Table 11, it can be seen that RioGNN still has obvious advantages compared with GAT and GraphSAGE, where AUC, Recall and F1 increase by 17.34%-28.53%, 2.64%-23.30%, 13.29%-20.90%. In addition, RioGNN has a relatively stable evaluation index among many variants. Among them, the results on Yelp dataset are compared with those with transductive learning in Table 5. In the inductive learning, the AUC and Recall rate of RioGNN constantly surpasses ROO-GNN variants although RioGNN is slightly lower than ROO-GNN in the transductive learning shown in Table 11. This represents the stability of the performance of the recursive framework in challenging tasks and illustrates its advantages in small-scale scenarios. In the Amazon dataset, due to the expansion of the data scale, some variants are better evaluated in some aspects, but RioGNN is stable at a relatively high level from the comprehensive situation of AUC, Recall and F1. This situation similarly appears in MIMIC-III. It is worth noting that the BIO-GNN variants in the Amazon and MIMIC-III datasets achieve a good performance improvement compared with their situation in the transductive learning task. We believe this is because BMAB has a relatively weak learning ability compared to Actor-Critic, which reduces the dependence of the learned model on the training set, so it is more compatible with newly added nodes. Overall, RioGNN has good applicability in both transductive learning and inductive learning.

5.5 Hyper-parameter Sensitivity

Figure 12 shows the test performance of the three hyper-parameters we introduce in Section 4.4 in three datasets. The first row of Figure 12 shows the AUC and Recall of the training set of RioGNN at different sampling ratios (note that the test results come from an unbalanced test set). It can be seen that when the sampling ratio is $1:0.2$, that is, when the negative samples are much smaller than the positive samples, overfitting occurs in all three datasets. Compared with $1:0.5$ and $1:2$ sampling ratios, $1:1$ sampling show higher AUC and Recall indicators in all three datasets. The second row of Figure 12 studies the backtracking structure we set in Section 3.2.2. From Figure 12(d), Figure 12(e) and Figure 12(f), models with backtracking settings bring stable performance in all datasets compared to models without backtracking. In the third row of Figure 12, we test different depth switching conditions. When the deep switching number is set to 3, AUC and Recall achieve good and balanced performance.



(a) Yelp (pos:neg). (b) Amazon (pos:neg). (c) MIMIC-III (pos:neg).

(d) Yelp. (e) Amazon. (f) MIMIC-III.

(g) Yelp. (h) Amazon. (i) MIMIC-III.

Fig. 12. Parameter sensitivity of under-sampling ratio, with & without backtracking, deep switching number on Yelp, Amazon and MIMIC-III.

6 RELATED WORK

In the past years, Graph Neural Networks (GNNs) and Reinforcement Learning (RL) technologies have received increasing attention and many upgraded algorithms have been proposed. Hence, the existing literature can be roughly classified into three categories: semi-supervised graph neural networks, RL, and RL-guided GNNs.

6.1 Semi-supervised Graph Neural Networks

According to the difference in data modeling for real-world graph data, we roughly divide the semi-supervised graph neural network methods into homogeneous graph neural networks, heterogeneous graph neural networks, and multiple graph learning models.

Homogeneous Graph Neural Networks. They are usually referred to those GNN methods that do not consider the data type of nodes or the attributes of the edges on the graph. Classical methods include GCN [49], Graph-SAGE [32] and GAT [96]. As discussed in the previous section, the GCN model defines the first successful graph convolutions in analogy to convolutional layers over Euclidean data, and is thus seen as a generalization of convolution neural networks (CNNs) for non-grid topologies. The Graph-SAGE model exploits sampling for obtaining a fixed number of neighbors for each node to generate node embedding via aggregation functions, which can be invariant if the permutations of node orderings, such as a mean, sum, or max function, are applied. Moreover, the Graph-SAGE model presents the first general inductive learning framework that continuously samples and aggregates its local neighbors' features to generate embedding for the new node. By contrast, the GAT model firstly adopts attention mechanisms to learn the relative weights between two connected nodes. The multi-head self-attention is further enforced to increase the model's expressive capability. Despite the powerful graph representation learning capability of these models, the main limitation is the ignorance of the diversity of data types and relationships manifesting in the real-world data and applications.

Heterogeneous Graph Neural Networks. Such approaches generally consider the heterogeneity of node types or edge types when aggregating feature information from node's local neighbors via neural networks. The classical meta-path and meta-graph based methods include GAS [53], HAN [101], Player2Vec [118], HSGNN [62] and MAGNN [22]. Considering the diversity of edges in real-world data, more relational graph neural network methods including R-GCN [85], SemiGNN [98], FdGars [99] and GraphConsis [61] are developed. Other heterogeneous graph neural networks, including HGT [41], GEM [60], HetSANN [38], etc., implement complex neural aggregations among heterogeneous neighbors. All these heterogeneous models are upgraded versions of the previous homogeneous models. Nevertheless, there is no literature exploring how to select neighbor nodes to build the most expressive, explanatory and stable aggregation.

Multiple Graph Learning Models. Apart from the above homogeneous and heterogeneous GNNs that solve single-graph representation learning, multi-graph neural network models [65, 66, 88, 102, 117] study fusing the multiple characterizes to comprehensively learn the embedding of graph data objects. MGAT [105] explores both attention-based architecture for learning node representations from each single view and view-focused attention method to aggregate the view-wise node representations. A multi-view knowledge graph embedding [115] is presented by using cross-view entity identity inference to capture the alignment information between two knowledge graphs. In order to filter out useless feature interactions, a Bayesian Personalized Feature Interaction Selection mechanism [8] is designed under the Bayesian Variable Selection (BVS) theory in recommendation tasks. Moreover, a block-diagonal regularization [9] is proposed to guide the item similarities in the top-N recommendation task.

6.2 Reinforcement Learning

With the development of technology, reinforcement learning algorithms have derived many different development directions. The more basic algorithms are value-based only Q-Learning [103] and DQN [69] algorithms, which use value functions to estimate and reduce the occurrence of local optimal situations. However, policy-based only algorithms such as PPO [86] directly performs iterative calculation on the policy, which can achieve better convergence. The Actor-Critic type of reinforcement learning methods AC [50], DDPG [55], TD3 [23], and SAC [36] combine the advantages of value-based and policy-based to train Q functions and strategies at the same time. In addition to the above division methods, from the perspective of action space types, DQN and Q-learning are suitable for discrete action spaces, DDPG, TD3, and SAC support continuous action spaces, while PPO and AC are suitable for both discrete and continuous action spaces. Or from the perspective of learning methods, reinforcement learning algorithms such as DDPG, DQN, SAC, and TD3 combine deep learning and use the fitting ability of neural networks to obtain better optimization. These different algorithms have different advantages, but

also bring different limitations. For example, algorithms that only support discrete action spaces are incapable of continuous action space requirements, Actor-Critic type algorithms have inherited the desired shortcomings while absorbing the value-based method and the policy-based method. The framework that only supports the same reinforcement learning algorithm has limitations in adaptability to many types of tasks.

6.3 Combination GNNs and RL

There are a few attempts to marry GNNs and RL. DGN+GNN [2] is a model used to generalize unseen network topologies, where GNNs that model the network environment allow the DRL agent to operate on different networks. G2S+BERT+RL [11] is a RL based graph-to-sequence model for natural question generation, where the answer information is utilized by an effective Deep Alignment Network and a novel bidirectional GNN is proposed to process the directed passage graph. Similarly, other work [34, 42, 90] investigates how to use GNNs to improve the generalization ability of RL. There are also numerous studies that leverage RL to optimize representation learning on graphs. For example, DeepPath [106] a knowledge graph embedding and reasoning framework based on RL policy-based; the RL agent is trained to ascertain the reasoning paths in the knowledge base. RL-HGNN [120] devises different meta-paths for any node in a HIN to learn its effective representations. It models the process of meta-path design as a Markov Decision Process by using a DRL-based policy network for adaptive meta-path selection. As opposed to RioGNN, the RL-HGNN model pays more attention to revealing meaningful meta-paths or relations in heterogeneous graph analysis. GraphNAS [24] employs a search space covering sampling functions, aggregation functions and gated functions and uses RL to search graph neural architectures. Policy-GNN [51] formulates the GNN training problem as a Markov Decision Process, and can adaptively learn an aggregation policy to sample diverse iterations of aggregations for different nodes. However, neither GraphNAS nor Policy-GNN models considers heterogeneous neighborhoods in aggregation although they pay more attention to neural architecture searching.

7 RELATED WORK

In the past years, Graph Neural Networks (GNNs) and Reinforcement Learning (RL) technologies have received increasing attention and many upgraded algorithms have been proposed. Hence, the existing literature can be roughly classified into three categories: semi-supervised graph neural networks, RL, and RL-guided GNNs.

7.1 Semi-supervised Graph Neural Networks

According to the difference in data modeling for real-world graph data, we roughly divide the semi-supervised graph neural network methods into homogeneous graph neural networks, heterogeneous graph neural networks, and multiple graph learning models.

Homogeneous Graph Neural Networks. They are usually referred to those GNN methods that do not consider the data type of nodes or the attributes of the edges on the graph. Classical methods include GCN [49], Graph-SAGE [32] and GAT [96]. As discussed in the previous section, the GCN model defines the first successful graph convolutions in analogy to convolutional layers over Euclidean data, and is thus seen as a generalization of convolution neural networks (CNNs) for non-grid topologies. The Graph-SAGE model exploits sampling for obtaining a fixed number of neighbors for each node to generate node embedding via aggregation functions, which can be invariant if the permutations of node orderings, such as a mean, sum, or max function, are applied. Moreover, the Graph-SAGE model presents the first general inductive learning framework that continuously samples and aggregates its local neighbors' features to generate embedding for the new node. By contrast, the GAT model firstly adopts attention mechanisms to learn the relative weights between two connected nodes. The multi-head self-attention is further enforced to increase the model's expressive capability. Despite the powerful

graph representation learning capability of these models, the main limitation is the ignorance of the diversity of data types and relationships manifesting in the real-world data and applications.

Heterogeneous Graph Neural Networks. Such approaches generally consider the heterogeneity of node types or edge types when aggregating feature information from node's local neighbors via neural networks. The classical meta-path and meta-graph based methods include GAS [53], HAN [101], Player2Vec [118], HSGNN [62] and MAGNN [22]. Considering the diversity of edges in real-world data, more relational graph neural network methods including R-GCN [85], SemiGNN [98], FdGars [99] and GraphConsis [61] are developed. Other heterogeneous graph neural networks, including HGT [41], GEM [60], HetSANN [38], etc., implement complex neural aggregations among heterogeneous neighbors. All these heterogeneous models are upgraded versions of the previous homogeneous models. Nevertheless, there is no literature exploring how to select neighbor nodes to build the most expressive, explanatory and stable aggregation.

Multiple Graph Learning Models. Apart from the above homogeneous and heterogeneous GNNs that solve single-graph representation learning, multi-graph neural network models [65, 66, 88, 102, 117] study fusing the multiple characterizes to comprehensively learn the embedding of graph data objects. MGAT [105] explores both attention-based architecture for learning node representations from each single view and view-focused attention method to aggregate the view-wise node representations. A multi-view knowledge graph embedding [115] is presented by using cross-view entity identity inference to capture the alignment information between two knowledge graphs. In order to filter out useless feature interactions, a Bayesian Personalized Feature Interaction Selection mechanism [8] is designed under the Bayesian Variable Selection (BVS) theory in recommendation tasks. Moreover, a block-diagonal regularization [9] is proposed to guide the item similarities in the top-N recommendation task.

7.2 Reinforcement Learning

With the development of technology, reinforcement learning algorithms have derived many different development directions. The more basic algorithms are value-based only Q-Learning [103] and DQN [69] algorithms, which use value functions to estimate and reduce the occurrence of local optimal situations. However, policy-based only algorithms such as PPO [86] directly performs iterative calculation on the policy, which can achieve better convergence. The Actor-Critic type of reinforcement learning methods AC [50], DDPG [55], TD3 [23], and SAC [36] combine the advantages of value-based and policy-based to train Q functions and strategies at the same time. In addition to the above division methods, from the perspective of action space types, DQN and Q-learning are suitable for discrete action spaces, DDPG, TD3, and SAC support continuous action spaces, while PPO and AC are suitable for both discrete and continuous action spaces. Or from the perspective of learning methods, reinforcement learning algorithms such as DDPG, DQN, SAC, and TD3 combine deep learning and use the fitting ability of neural networks to obtain better optimization. These different algorithms have different advantages, but also bring different limitations. For example, algorithms that only support discrete action spaces are incapable of continuous action space requirements, Actor-Critic type algorithms have inherited the desired shortcomings while absorbing the value-based method and the policy-based method. The framework that only supports the same reinforcement learning algorithm has limitations in adaptability to many types of tasks.

7.3 Combination GNNs and RL

There are a few attempts to marry GNNs and RL. DGN+GNN [2] is a model used to generalize unseen network topologies, where GNNs that model the network environment allow the DRL agent to operate on different networks. G2S+BERT+RL [11] is a RL based graph-to-sequence model for natural question generation, where the answer information is utilized by an effective Deep Alignment Network and a novel bidirectional GNN is proposed to process the directed passage graph. Similarly, other work [34, 42, 90] investigates how to use

GNNs to improve the generalization ability of RL. There are also numerous studies that leverage RL to optimize representation learning on graphs. For example, DeepPath [106] a knowledge graph embedding and reasoning framework based on RL policy-based; the RL agent is trained to ascertain the reasoning paths in the knowledge base. RL-HGNN [120] devises different meta-paths for any node in a HIN to learn its effective representations. It models the process of meta-path design as a Markov Decision Process by using a DRL-based policy network for adaptive meta-path selection. As opposed to RioGNN, the RL-HGNN model pays more attention to revealing meaningful meta-paths or relations in heterogeneous graph analysis. GraphNAS [24] employs a search space covering sampling functions, aggregation functions and gated functions and uses RL to search graph neural architectures. Policy-GNN [51] formulates the GNN training problem as a Markov Decision Process, and can adaptively learn an aggregation policy to sample diverse iterations of aggregations for different nodes. However, neither GraphNAS nor Policy-GNN models considers heterogeneous neighborhoods in aggregation although they pay more attention to neural architecture searching.

8 CONCLUSION AND FUTURE WORK

This paper studies RioGNN, a reinforced, recursive and flexible neighborhood selection guided multi-relational Graph Neural Network architecture, to learn more discriminative node embedding and respond to the explanation of the importance of different relations in spam review detection and disease diagnosis tasks, respectively. RioGNN designs a label-aware neural similarity neighbor measure and reinforced relation-aware neighbor selectors using reinforcement learning technology, respectively. To optimize the computational efficiency of the reinforcement neighbor selecting, we further design a recursive and scalable framework with estimable depth and width for different scales of multi-relational graphs. The conducted experiments on three real-world benchmark datasets suggest that RioGNN significantly, consistently and steadily outperforms the state-of-the-art alternatives across all the datasets. Our work shows the promise in learning a reinforced neighborhood aggregation for GNNs, potentially opening new avenues for future research in boosting the performance of GNNs with adaptive neighborhood selection and analysing the importance of different relations in message passing.

In the future, we aim to adopt a multi-agent RL algorithm to further enable the RioGNN to adaptively identify meaningful relations for each node, instead of the manual efforts in defining relations, for automated representation learning on heterogeneous data. In addition, it is also interesting to study how to extend our models to other tasks on graph data analysis and application, such as the personalized recommendation system, social network analysis, and etc.

ACKNOWLEDGMENT

The authors of this paper are supported by NSFC through grants 62002007 and U20B2053, S&T Program of Hebei through grant 20310101D, Fundamental Research Funds for the Central Universities. Renyu Yang is partially supported by UK EPSRC Grant (EP/T01461X/1) and UK White Rose University Consortium. Philip S. Yu is partially supported by NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941.

REFERENCES

[1] Aisha Abdallah, Mohd Aizaini Maarof, and Anazida Zainal. 2016. Fraud detection system: A survey. *Journal of Network and Computer Applications* 68 (2016), 90–113.

[2] Paul Almasan, José Suárez-Varela, Arnau Badia-Sampera, Krzysztof Rusek, Pere Barlet-Ros, and Albert Cabellos-Aparicio. 2019. Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *arXiv preprint arXiv:1910.07421* (2019).

[3] Victor Bapst, Thomas Keck, A Grabska-Barwińska, Craig Donner, Ekin Dogus Cubuk, Samuel S Schoenholz, Annette Obika, Alexander WR Nelson, Trevor Back, Demis Hassabis, et al. 2020. Unveiling the predictive power of static structure in glassy systems. *Nature Physics* 16, 4 (2020), 448–454.

[4] Bokai Cao, Mia Mao, Siim Viidu, and Philip S. Yu. 2017. HitFraud: a broad learning approach for collective fraud detection in heterogeneous information networks. In *Proceedings of the IEEE ICDM*. 769–774.

[5] Yuwei Cao, Hao Peng, Jia Wu, Yingtong Dou, Jianxin Li, and Philip S. Yu. 2021. Knowledge-Preserving Incremental Social Event Detection via Heterogeneous GNNs. In *Proceedings of the Web Conference 2021*. Association for Computing Machinery, 3383–3395.

[6] Yuwei Cao, Hao Peng, and Philip S. Yu. 2020. Multi-information Source HIN for Medical Concept Embedding. In *Proceedings of the PAKDD*. Springer, 396–408.

[7] Hao Chen, Yue Xu, Feiran Huang, Zengde Deng, Wenbing Huang, Senzhang Wang, Peng He, and Zhoujun Li. 2020. Label-Aware Graph Convolutional Networks. In *Proceedings of the CIKM*. ACM, 1977–1980.

[8] Yifan Chen, Pengjie Ren, Yang Wang, and Maarten de Rijke. 2019. Bayesian Personalized Feature Interaction Selection for Factorization Machines. In *Proceedings of ACM SIGIR*. 665–674.

[9] Yifan Chen, Yang Wang, Xiang Zhao, Jie Zou, and Maarten De Rijke. 2020. Block-Aware Item Similarity Models for Top-N Recommendation. *ACM Transactions on Information Systems* 38, 4, Article 42 (2020), 26 pages.

[10] Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2019. Deep iterative and adaptive learning for graph neural networks. *arXiv preprint arXiv:1912.07832* (2019).

[11] Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2019. Reinforcement Learning Based Graph-to-Sequence Model for Natural Question Generation. In *Proceedings of the ICLR*.

[12] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the SIGKDD*. ACM, 257–266.

[13] Edward Choi, Zhen Xu, Yujia Li, Michael Dusenberry, Gerardo Flores, Emily Xue, and Andrew Dai. 2020. Learning the graphical structure of electronic health records with graph convolutional transformer. In *Proceedings of the AAAI*, Vol. 34. 606–613.

[14] Yingtong Dou. 2019. A review of recent advance in online spam detection. (2019).

[15] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S. Yu. 2020. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the ACM CIKM*. 315–324.

[16] Yingtong Dou, Guixiang Ma, Philip S. Yu, and Sihong Xie. 2020. Robust Spammer Detection by Nash Reinforcement Learning. In *Proceedings of the SIGKDD*. ACM, 924–933.

[17] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. 2016. Deep Reinforcement Learning in Large Discrete Action Spaces. arXiv:1512.07679

[18] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of the NIPS*. 2224–2232.

[19] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *Proceedings of the Web Conference*. ACM, 417–426.

[20] Meherwar Fatima, Maruf Pasha, et al. 2017. Survey of machine learning algorithms for disease diagnostic. *Journal of Intelligent Learning Systems and Applications* 9, 01 (2017), 1.

[21] Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of the aaai conference on artificial intelligence*, Vol. 32.

[22] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *Proceedings of the WWW*. ACM, 2331–2341.

[23] Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, Stockholmsmässan, Stockholm Sweden, 1587–1596.

[24] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2019. Graphnas: Graph neural architecture search with reinforcement learning. *arXiv preprint arXiv:1904.09981* (2019).

[25] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the ICML*. PMLR, 1263–1272.

[26] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E. Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation* 101, 23 (2000), E215.

[27] Jibing Gong, Shen Wang, Jinlong Wang, Wenzheng Feng, Hao Peng, Jie Tang, and Philip S. Yu. 2020. Attentional graph convolutional networks for knowledge concept recommendation in moocs in a heterogeneous view. In *Proceedings of the ACM SIGIR*. 79–88.

[28] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677* (2017).

[29] Steve Gregory. 2010. Finding overlapping communities in networks by label propagation. *New journal of Physics* 12, 10 (2010), 103018.

[30] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI*, Vol. 33. AAAI Press, 922–929.

[31] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905* (2018).

[32] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[33] Bowen Hao, Jing Zhang, Hongzhi Yin, Cuiping Li, and Hong Chen. 2021. Pre-Training Graph Neural Networks for Cold-Start Users and Items Representation. In *Proceedings of the WSDM*. ACM, 265–273.

[34] Patrick Hart and Alois Knoll. 2020. Graph Neural Networks and Reinforcement Learning for Behavior Generation in Semantic Environments. In *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1589–1594.

[35] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double Q-Learning. In *Proceedings of the AAAI*. AAAI Press, 2094–2100.

[36] Yu He, Yangqiu Song, Jianxin Li, Cheng Ji, Jian Peng, and Hao Peng. 2019. HeteSpaceyWalk: a heterogeneous spacey random walk for heterogeneous information network embedding. In *Proceedings of the CIKM*. ACM, 639–648.

[37] Yiming Hei, Renyu Yang, Hao Peng, Lihong Wang, Xiaolin Xu, Jianwei Liu, Hong Liu, Jie Xu, and Lichao Sun. 2021. Hawk: Rapid android malware detection through heterogeneous graph attention networks. *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[38] Huiting Hong, Hantao Guo, Yucheng Lin, Xiaoqing Yang, Zang Li, and Jieping Ye. 2020. An attention-based graph neural network for heterogeneous structural learning. In *Proceedings of the AAAI*. 4132–4139.

[39] Anahita Hosseini, Ting Chen, Wenjun Wu, Yizhou Sun, and Majid Sarrafzadeh. 2018. Heteromed: Heterogeneous information network for medical diagnosis. In *Proceedings of the CIKM*. ACM, 763–772.

[40] Junling Hu and Michael P. Wellman. 1998. Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 242–250.

[41] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of the WWW*. ACM, 2704–2710.

[42] Jaromír Janisch, Tomáš Pevnỳ, and Viliam Lisỳ. 2020. Symbolic Relational Deep Reinforcement Learning based on Graph Neural Networks. *arXiv preprint arXiv:2009.12462* (2020).

[43] Di Jin, Ziyang Liu, Weihao Li, Dongxiao He, and Weixiong Zhang. 2019. Graph convolutional networks meet markov random fields: Semi-supervised community detection in attribute networks. In *Proceedings of the AAAI*. AAAI Press, 152–159.

[44] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3, 1 (2016), 1–9.

[45] Alistair E. W Johnson, Tom J Pollard, Lu Shen, Li Wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. [n.d.]. MIMIC-III, a freely accessible critical care database. *Scientific Data* ([n. d.]).

[46] P. Kaghazgaran, J. Caverlee, and A. Squicciarini. 2018. Combating crowdsourced review manipulators: A neighborhood-based approach. In *Proceedings of the WSDM*. ACM, 306–314.

[47] Anssi Kanervisto, Christian Scheller, and Ville Hautamäki. 2020. Action Space Shaping in Deep Reinforcement Learning. In *2020 IEEE Conference on Games (CoG)*. 479–486.

[48] Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Proceedings of the NIPS*. 4284–4295.

[49] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *Proceedings of the ICLR*.

[50] Vijay R Konda and John N Tsitsiklis. 2000. Actor-critic algorithms. In *Proceedings of the NIPS*. 1008–1014.

[51] Kwei-Herng Lai, Daochen Zha, Kaixiong Zhou, and Xia Hu. 2020. Policy-GNN: Aggregation Optimization for Graph Neural Networks. In *Proceedings of the ACM SIGKDD*. New York, NY, USA, 461–471.

[52] Kai Lei, Meng Qin, Bo Bai, Gong Zhang, and Min Yang. 2019. GCN-GAN: A non-linear temporal link prediction model for weighted dynamic networks. In *Proceedings of the IEEE INFOCOM*. 388–396.

[53] Ao Li, Zhou Qin, Runshi Liu, Yiqun Yang, and Dong Li. 2019. Spam review detection with graph convolutional networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 2703–2711.

[54] Ruoyu Li and Sheng Wang. 2018. Adaptive Graph Convolutional Neural Networks. In *Proceedings of the AAAI*. AAAI Press, 3546–3553.

[55] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2019. Continuous control with deep reinforcement learning. arXiv:1509.02971 [cs.LG]

[56] Weiyang Liu, Zhen Liu, James M Rehg, and Le Song. 2019. Neural similarity learning. In *Proceedings of the NIPS*. 5025–5036.

[57] Xin Liu and Tsuyoshi Murata. 2010. Advanced modularity-specialized label propagation algorithm for detecting communities in networks. *Physica A: Statistical Mechanics and its Applications* 389, 7 (2010), 1493–1500.

[58] Ye Liu, Yao Wan, Lifang He, Hao Peng, and Yu Philip S. 2021. KG-BART: Knowledge Graph-Augmented BART for Generative Commonsense Reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 6418–6425.

[59] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. 2019. GeniePath: Graph Neural Networks with Adaptive Receptive Paths. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 4424–4431.

[60] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. 2018. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2077–2085.
[61] Zhiwei Liu, Yingtong Dou, Philip S. Yu, Yutong Deng, and Hao Peng. 2020. Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection. *Proceedings of the SIGIR*, 1569–1572.
[62] Zheng Liu, Xiaohan Li, Hao Peng, Lifang He, and Philip S. Yu. 2020. Heterogeneous Similarity Graph Neural Network on Electronic Health Records. In *Proceedings of the BigData*. IEEE.
[63] Chen Lu, Chen Zhi, Tan Bowen, Long Sishan, Gašić Milica, and Yu Kai. 2019. AgentGraph: Toward Universal Dialogue Management With Structured Deep Reinforcement Learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27, 9 (2019), 1378–1391.
[64] Jake Luo, Christina Eldredge, Chi C Cho, and Ron A Cisler. 2016. Population Analysis of Adverse Events in Different Age Groups Using Big Clinical Trials Data. *JMIR Med Inform* 4, 4 (17 Oct 2016), e30. https://doi.org/10.2196/medinform.6437
[65] Guixiang Ma, Lifang He, Chun-Ta Lu, Weixiang Shao, Philip S. Yu, Alex D Leow, and Ann B Ragin. 2017. Multi-view clustering with graph embedding for connectome analysis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 127–136.
[66] Tengfei Ma, Cao Xiao, Jiayu Zhou, and Fei Wang. 2018. Drug similarity integration through attentive multi-view graph auto-encoders. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3477–3483.
[67] Julian John McAuley and Jure Leskovec. 2013. From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews. In *Proceedings of the 22nd International Conference on World Wide Web* (Rio de Janeiro, Brazil) *(WWW '13)*. Association for Computing Machinery, New York, NY, USA, 897–908.
[68] Changping Meng, Reynold Cheng, Silviu Maniu, Pierre Senellart, and Wangda Zhang. 2015. Discovering meta-paths in large heterogeneous information networks. In *Proceedings of the WWW*. ACM, 754–764.
[69] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.
[70] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. 2013. What Yelp Fake Review Filter Might Be Doing? *Proceedings of the International AAAI Conference on Web and Social Media* 7, 1 (Jun. 2013).
[71] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs. In *Proceedings of the ACL*. ACL, 4710–4723.
[72] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially regularized graph autoencoder for graph embedding. In *Proceedings of the IJCAI*. AAAI Press, 2609–2615.
[73] Hao Peng, Haoran Li, Yangqiu Song, Vincent Zheng, and Jianxin Li. 2021. Federated Knowledge Graphs Embedding. In *Proceedings of the ACM CIKM*. 1–10.
[74] Hao Peng, Jianxin Li, Qiran Gong, Yuanxin Ning, Senzhang Wang, and Lifang He. 2020. Motif-Matching Based Subgraph-Level Attentional Convolutional Network for Graph Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5387–5394.
[75] Hao Peng, Jianxin Li, Qiran Gong, Yangqiu Song, Yuanxing Ning, Kunfeng Lai, and Philip S. Yu. 2019. Fine-grained event categorization with heterogeneous graph convolutional networks. In *Proceedings of the IJCAI*. AAAI Press, 3238–3245.
[76] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the WWW*. 1063–1072.
[77] Hao Peng, Jianxin Li, Yangqiu Song, Renyu Yang, Ranjan Rajiv, Philip S. Yu, and He Lifang. 2021. Streaming Social Event Detection and Evolution Discovery in Heterogeneous Information Networks. *ACM Transactions on Knowledge Discovery from Data* 15, 5 (2021).
[78] Hao Peng, Jianxin Li, Senzhang Wang, Lihong Wang, Qiran Gong, Renyu Yang, Bo Li, Philip S. Yu, and Lifang He. 2021. Hierarchical taxonomy-aware and attentional graph capsule RCNNs for large-scale multi-label text classification. *IEEE Transactions on Knowledge and Data Engineering* 33, 6 (2021), 2505–2519.
[79] Hao Peng, Jianxin Li, Zheng Wang, Renyu Yang, Mingzhe Liu, Mingming Zhang, Philip S Yu, and Lifang He. 2021. Lifelong Property Price Prediction: A Case Study for the Toronto Real Estate Market. *IEEE Transactions on Knowledge and Data Engineering* (2021).
[80] Hao Peng, Hongfei Wang, Bowen Du, Md Zakirul Alam Bhuiyan, Hongyuan Ma, Jianwei Liu, Lihong Wang, Zeyu Yang, Linfeng Du, Senzhang Wang, et al. 2020. Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting. *Information Sciences* 521 (2020), 277–290.
[81] Hao Peng, Renyu Yang, Zheng Wang, Jianxin Li, Lifang He, Philip Yu, Albert Zomaya, and Raj Ranjan. 2021. Lime: Low-cost incremental learning for dynamic heterogeneous information networks. *IEEE Trans. Comput.* (2021).
[82] D. Pozo and J. Contreras. 2011. Finding Multiple Nash Equilibria in Pool-Based Markets: A Stochastic EPEC Approach. *IEEE Transactions on Power Systems* 26, 3 (2011), 1744–1752.
[83] Ruihong Qiu, Zi Huang, Jingjing Li, and Hongzhi Yin. 2020. Exploiting Cross-session Information for Session-based Recommendation with Graph Neural Networks. *ACM Transactions on Information Systems (TOIS)* 38, 3 (2020), 1–23.

[84] Shebuti Rayana and Leman Akoglu. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the SIGKDD*. ACM, 985–994.
[85] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of the ESWC*. Springer, 593–607.
[86] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
[87] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. 2016. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2016), 17–37.
[88] Junkai Sun, Junbo Zhang, Qiaofei Li, Xiuwen Yi, Yuxuan Liang, and Yu Zheng. 2020. Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks. *IEEE Transactions on Knowledge and Data Engineering* (2020).
[89] Lichao Sun, Yingtong Dou, Carl Yang, Ji Wang, Philip S. Yu, Lifang He, and Bo Li. 2018. Adversarial Attack and Defense on Graph Data: A Survey. *arXiv preprint arXiv:1812.10528* (2018).
[90] Penghao Sun, Julong Lan, Junfei Li, Zehua Guo, and Yuxiang Hu. 2020. Combining deep reinforcement learning with graph neural networks for optimal VNF placement. *IEEE Communications Letters* 25, 1 (2020), 176–180.
[91] Qingyun Sun, Hao Peng, Jianxin Li, Senzhang Wang, Xiangyu Dong, Liangxuan Zhao, Philip S. Yu, and Lifang He. 2020. Pairwise Learning for Name Disambiguation in Large-Scale Heterogeneous Academic Networks. In *Proceedings of the IEEE ICDM*. IEEE, 511–520.
[92] Qingyun Sun, Hao Peng, Jianxin Li, Jia Wu, Yuanxing Ning, Phillip S. Yu, and Lifang He. 2021. SUGAR: Subgraph Neural Network with Reinforcement Pooling and Self-Supervised Mutual Information Mechanism. In *Proceedings of the Web Conference*. 2081–2091.
[93] Yingcheng Sun and Kenneth Loparo. 2019. Opinion spam detection based on heterogeneous information network. In *Proceedings of the IEEE ICTAI*. 1156–1163.
[94] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
[95] Fawcett Tom. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (2006), 861–874.
[96] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *Proceedings of the ICLR*.
[97] Vikas Verma, Meng Qu, Alex Lamb, Yoshua Bengio, Juho Kannala, and Jian Tang. 2019. Graphmix: Regularized training of graph neural networks for semi-supervised learning. *arXiv preprint arXiv:1909.11715* (2019).
[98] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. 2019. A Semi-supervised Graph Attentive Network for Financial Fraud Detection. In *Proceedings of the IEEE ICDM*. 598–607.
[99] Jianyu Wang, Rui Wen, Chunming Wu, Yu Huang, and Jian Xion. 2019. Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *Proceedings of the World Wide Web Conference*. 310–316.
[100] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and Philip S. Yu. 2020. A Survey on Heterogeneous Graph Embedding: Methods, Techniques, Applications and Sources. *arXiv preprint arXiv:2011.14867* (2020).
[101] Xiao Wang, Houye Ji, Chuan Shi, et al. 2019. Heterogeneous graph attention network. In *Proceedings of the Web Conference*. ACM, 2022–2032.
[102] Yang Wang. 2021. Survey on Deep Multi-Modal Data Analytics: Collaboration, Rivalry, and Fusion. *ACM Transactions on Multimedia Computing, Communications, and Applications* 17, 1s, Article 10 (2021), 25 pages.
[103] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.
[104] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).
[105] Yu Xie, Yuanqiao Zhang, Maoguo Gong, Zedong Tang, and Chao Han. 2020. Mgat: Multi-view graph attention networks. *Neural Networks* 132 (2020), 180–189.
[106] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *Proceedings of the EMNLP*. ACL, 564–573.
[107] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks?. In *Proceedings of the ICLR*.
[108] Carl Yang, Yichen Feng, Pan Li, Yu Shi, and Jiawei Han. 2018. Meta-graph based hin spectral embedding: Methods, analyses, and insights. In *Proceedings of the IEEE ICDM*. 657–666.
[109] Yiying Yang, Xi Yin, Haiqin Yang, Xingjian Fei, Hao Peng, Kaijie Zhou, Kunfeng Lai, and Jianping Shen. 2021. KGSynNet: A Novel Entity Synonyms Discovery Framework with Knowledge Graph. In *Proceedings of DASFAA*. Springer International Publishing, Cham, 174–190.
[110] Selim F Yilmaz and Suleyman S Kozat. 2020. Unsupervised Anomaly Detection via Deep Metric Learning with End-to-End Optimization. *arXiv preprint arXiv:2005.05865* (2020).
[111] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the NIPS*. 4800–4810.

[112] Chen Yu, Wu Lingfei, and J. Zaki Mohammed. 2020. Deep Iterative and Adaptive Learning for Graph Neural Networks. *AAAI Workshops* (2020).
[113] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the SIGKDD*. ACM, 793–803.
[114] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *Proceedings of the NIPS*. 5165–5175.
[115] Qingheng Zhang, Zequn Sun, Wei Hu, Muhao Chen, Lingbing Guo, and Yuzhong Qu. 2019. Multi-view Knowledge Graph Embedding for Entity Alignment. In *Proceedings of the IJCAI*. 5429–5435.
[116] Shijie Zhang, Hongzhi Yin, Tong Chen, Quoc Viet Nguyen Hung, Zi Huang, and Lizhen Cui. 2020. *GCN-Based User Representation Learning for Unifying Robust Recommendation and Fraudster Detection.* Association for Computing Machinery, New York, NY, USA, 689–698.
[117] Xi Zhang, Lifang He, Kun Chen, Yuan Luo, Jiayu Zhou, and Fei Wang. 2018. Multi-view graph convolutional network and its applications on neuroimage analysis for parkinson's disease. In *AMIA Annual Symposium Proceedings*, Vol. 2018. 1147.
[118] Yiming Zhang, Yujie Fan, Yanfang Ye, Liang Zhao, and Chuan Shi. 2019. Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework. In *Proceedings of the CIKM*. ACM, 549–558.
[119] H. Zheng, M. Xue, H. Lu, S. Hao, H. Zhu, X. Liang, and K. Ross. 2018. Smoke screener or straight shooter: Detecting elite sybil attacks in user-review social networks. *Proceedings of the NDSS*.
[120] Zhiqiang Zhong, Cheng-Te Li, and Jun Pang. 2020. Reinforcement Learning Enhanced Heterogeneous Graph Neural Network. *arXiv preprint arXiv:2010.13735* (2020).
[121] Shichao Zhu, Chuan Zhou, Shirui Pan, Xingquan Zhu, and Bin Wang. 2019. Relation structure-aware heterogeneous graph neural network. In *Proceedings of the IEEE ICDM*. 1534–1539.
[122] Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. (2002).
[123] Marinka Zitnik and Jure Leskovec. 2017. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* 33, 14 (2017), i190–i198.

Received April 2021; Revised August 2021; Accepted September 2021