RAHG: A Role-Aware Hypergraph Neural Network for Node Classification in Graphs

Kunhao Li, Zhenhua Huang, and Zhaohong Jia, *Member, IEEE*

***Abstract*—Graph neural networks have been widely studied and applied to node classification in graph-format datasets in recent years. Traditional graph neural networks mainly consider the adjacency characteristics of nodes and fail to learn rich role representations of nodes. Existing role representations methods of nodes are mostly in unsupervised approaches, resulting in unsatisfactory performance in downstream tasks. A graph can be reorganized as a hypergraph, in which the role characteristics of nodes are more intuitively represented. Based on this, we propose a role-aware hypergraph neural network (RAHG) that utilizes hypergraphs and an attention mechanism to fuse nodes' role and adjacency representations. A residual network is also applied to relieve the smoothing problem between layers in the model. The model adjusts the weights on the role and adjacency representations according to the characteristics of the graphs. RAHG significantly improves the prediction performance compared with existing graph neural networks on seven datasets, with accuracy increased by up to 12.1% on the node classification task in graphs.**

***Index Terms*—Graph neural networks, graph representation, node classification, hypergraph.**

I. Introduction

MANY real-world datasets are represented in graph format, e.g., interactions between users in social networks[1], [2], bundling relationships between users and communities in e-commerce[3], [4], sensor connections in the Internet of Things [5], and molecular networks [6], [7], etc. Graph neural networks (GNNs) have advantageous performance compared with traditional neural networks on many applications [8], [9]. Classic GNNs include GCN [10], GAT [11], GraphSAGE[12], VGAE[13] and GIN [14], etc. GNNs learn the nature of spatial features in graph datasets and improve the node and graph representations of neural networks. Most GNNs adopt the basic idea that aggregates neighboring information based on the message passing [15], which is based on gathering neighboring or adjacent node features in graphs.

However, nodes have different social roles or functionalities, and the nodes with similar roles or functionalities are not always

Manuscript received 1 September 2022; revised 2 January 2023; accepted 5 February 2023. Date of publication 7 February 2023; date of current version 16 June 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 71971002, in part by the Natural Science Foundation of Guangdong Province, China under Grant 2019A1515011792, and in part by the Natural Science Foundation of Anhui Province, China under Grant 1908085QF283. Recommended for acceptance by Yang Xiao. *(Kunhao Li and Zhenhua Huang contributed equally to this work.) (Corresponding authors: Zhenhua Huang; Zhaohong Jia.)*

The authors are with the School of Internet, Anhui University, Hefei 230061, China (e-mail: kunhomlihf@gmail.com; zhhuangscut@gmail.com; zhjia@mail.ustc.edu.cn).

Digital Object Identifier 10.1109/TNSE.2023.3243058

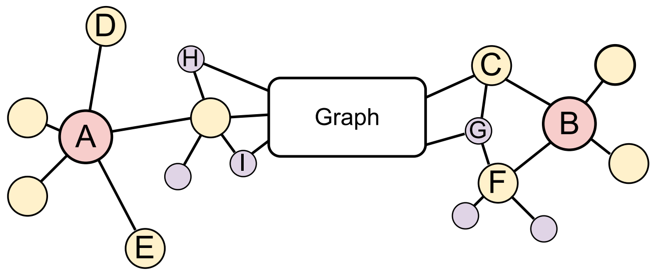

Fig. 1. A toy example graph. Nodes with the same color have similar roles. Nodes A and B, C and D are not directly connected. It is difficult for traditional GNNs to build a relationship between these nodes with similar roles.

directly connected, for instance, nodes A and B, D and E in Fig. 1. Traditional GNNs (including GCN [10], GAT [11], GraphSAGE [12], etc.) obtain the feature embedding of central nodes by aggregating the features of neighbor nodes, which produces representations containing information of local subgraphs. Nevertheless, graph aggregation in traditional GNNs has two limitations: (1) Hard to produce node representations that contain rich role characteristics; (2) Difficult to build a relationship between nodes with similar roles far away or not directly connected. The role of a node can be its semantic meanings in the real world, e.g., student, advisor, and manager, or structural roles in networks, e.g., structural hole spanner, opinion leaders [16]. Role representation learning of nodes has been explored in unsupervised ways [17], [18], [19]. However, due to the limited performance of unsupervised methods, they produce unsatisfactory results in downstream tasks [12]. Moreover, multi-layer GNNs based on adjacency suffer from the over-smoothing problems that nodes share similar features after several message passing layers.

The inputs of traditional GNNs are mostly typical graphs, which reflect adjacency relationships of nodes and ignore role relationships between nodes. On the contrary, hypergraphs [20] can be constructed in various ways based on the typical graph structure. The hypergraph adjacency contains more prosperous relations of nodes and clusters, making aggregation computations in graph neural networks more effective in the role and adjacency representations.

To address the limitations mentioned above of graph neural networks, inspired by the hypergraph [20], we propose a **R**ole-**A**ware **H**yper**G**raph Neural Network (RAHG) that considers role representations into graph neural networks. RAHG aggregates node representations of role and adjacency from high-order perspectives under hypergraph construction methods. We design two approaches to construct hypergraphs, which aggregate node information from the degree and neighbor levels. The nodes with similar roles are more closely connected in hypergraphs. RAHG applies an attention mechanism with learnable weights to combine the nodes' role and adjacency representations. By hypergraph convolution layers, the structure of hypergraphs is learned in the training process. Furthermore, the residual network is also applied to relieve the smoothing problem between hypergraph convolution layers. Compared with traditional GNNs, RAHG adds role characteristics and utilizes a more efficient aggregation method. RAHG is suitable for any node representation task with adjacency structure and role information. The main contributions are as follows[1]:

- We propose a novel role-aware hypergraph neural network RAHG by introducing an attention mechanism and hypergraph convolution layers that aggregate role and adjacency representations of nodes to address the limitations of current graph neural networks. The residual network is also applied in RAHG to alleviate the over-smoothing problem effectively.
- Two hypergraph construction approaches based on node links and node degrees are proposed. Both approaches gather nodes with similar roles closer to high-order perspectives and benefit role representation learning in GNNs.
- A new real-world graph dataset that records the relationships between advisors and graduate students in a university is built, which can be used to verify the performance in the node classification task.
- Extensive experiments on the new dataset and the other four public datasets demonstrate that RAHG highly improves the accuracy of node classification by large margins up to 12.1 (19.9% relative improvement).

The structure of this paper is organized as follows. In Section II, we review related works about graph neural networks, graph role representation, and hypergraph neural networks. Section III defines the problem of node classification and explains role and adjacency embeddings. In Section IV, details of the proposed model are introduced. Extensive experiments and analysis are represented in Section V. We summarize the works in Section VI.

II. Related Works

Recent works related to our research, including graph neural networks, graph role representation, and hypergraph neural networks are briefly introduced in this section.

A. Graph Neural Network

The graph neural network is a kind of neural network based on message-passing applied to the graph structure. GCN [10] and ChebyNet [21] apply neighbors propagation to aggregate the adjacency information. The GAT [11] introduces an attention mechanism to aggregate neighboring features. GraphSAGE [12] extends GCN to an inductive framework and makes it suitable for unseen nodes. LightGCN [22] learns user and item embeddings by linearly propagating features across the user-item interaction graph. Yang et al. [23] unified the hyperbolic graph neural networks into a general framework and summarized the variants of each component. SAT [24] exploits existing GNNs to extract subgraph representations and improve performance relative to the GNNs. Feature aggregation methods of these GNNs are based on adjacency message-passing.

Besides adjacency feature aggregations, some graph neural networks consider nodes' location or identity characteristics not applied in the above methods. ID-GNN [25] inductively considers node identities to embed node features in the message-passing process. The location-aware graph neural network (P-GNN) [26] is proposed for computing location-aware node embeddings. Li et al. [27] proposed GLOGNN and GLOGNN++ to learn a coefficient matrix to capture the correlations between nodes and perform neighborhood aggregation based on it. Xie et al. [28] proposed an end-to-end scale-aware graph neural network (SAGNN) by reasoning about cross-scale relationships between query images supported by few-shot semantic segmentation. Xu et al. [29] proposed a Graph wavelet Neural network (GWNN), which uses graph wavelet transform to address the shortcomings of previous spectral graph CNNs that rely on graph Fourier transform. These GNNs achieve efficacious results but do not explicitly address the relationship between higher-order aggregation and role information.

B. Graph Role Representation

Graph representation (i.e., Graph Embedding or Network Embedding) is a task of mapping nodes to low-dimension vectors that keep the structural information in original graphs [30]. Deepwalk [31] and Node2vec [32] proposed a random walk strategy and feed the generated node sequences into Skip-Gram [33] to produce the adjacency-based embeddings. LINE [34] is an edge-sampling algorithm that improves the effectiveness and efficiency of network embedding. The design of SDNE [35] extends LINE and optimizes nodes' first-order and second-order similarities by deep learning-based methods. Ying et al. [36] proposed the Graphormer to better model graph-structured data by the Transformer. LCNN [37] builds receptive fields by learning the position of each node based on structural and feature information. In role representations of nodes in graphs, Struc2vec [38] applies a hierarchical structure to measure node similarity at different scales and builds a multi-layer graph to encode adjacency similarity. Although the role characteristics of nodes are reflected, the computation of the multi-layer graph is time-consuming. Graphwave [19] represents the role of nodes through a heat wavelet diffusion.

C. Hypergraph Neural Network

Hypergraph learning was introduced in [39] as a propagation process on hypergraph structure. Compared with the traditional simple graph, a hypergraph can encode high-order data correlation (beyond pairwise connections) using its degree-free hyperedges [40]. Feng et al. [20] and Yadati et al. [40] incorporated hypergraphs into learning graph neural networks for the

[1]All the codes and datasets are published on https://github.com/PreckLi/RAHG.

TABLE I
NOTATIONS IN RAHG.

Symbols	Definition and description
G	A general undirected graph
G_H^l,G_H^d	Hypergraphs constructed by node link, and node degree
E_l,E_d	Hyperedge by node link, and node degree
W_l,W_d	Hyperedge weight matrices by node link, and node degree
D_e^l,D_e^d	Hyperedge degree matrices by node link, and node degree
D_v^l,D_v^d	Node degree matrices by node link, and node degree
X_r,X_a	Role embeddings, and adjacency embeddings
Z	The embedding of nodes
Y	The label of nodes

first time. Bai et al. [41] proposed a hypergraph convolutional neural network and a hypergraph attention network to enhance the capacity of representation learning. Ji et al. [42] proposed a two-channel hypergraph convolutional network framework DHCF for the collaborative filtering of recommend systems. Chien et al. [43] proposed a generalization framework for hypergraph neural networks, which adaptively learns the propagation mode suitable for data. Yang et al. [44] proposed an attention hypergraph neural network that applies an attention mechanism to filter essential attributes. Hypergraph neural networks have been developed and applied in different domains. However, they are not used to deal with node representations containing rich role features to our best knowledge.

III. Preliminaries

This section gives the problem definition of RAHG and the specific explanation of role and structure embedding.

A. Problem Description

An undirected graph is denoted as $G = (V, E, A)$, where $V = \{v_1, v_2, \ldots, v_N\}$ denotes the node set, E denotes the edge set, and $A \in \mathcal{R}^{N\times N}$ denotes the adjacency matrix. $X_r \in \mathcal{R}^{N\times F_r}$ is the role representation, $X_a \in \mathcal{R}^{N\times F_a}$ is the adjacency representation. $Y \in \mathcal{R}^N$ is the label of nodes with M categories. The GNNs produce nodes representation $Z \in \mathcal{R}^{N\times F_h}$ and predict the nodes' label. The symbols used in this paper and their definitions are summarized in Table I.

B. Role and Adjacency Embedding

The initial roles and adjacency embeddings of nodes generated by the graph structure through wavelet diffusion [19] and random walk [32] are used as input features of RAHG. This subsection describes the generation process for both types of embeddings.

1) Role Embedding: For a given graph's Laplacian matrix $L = D - A = U\Lambda U^T$, The corresponding spectral graph wavelet can be calculated as $\Psi = Ue^{-s\Lambda}U^T$, where the column corresponding to node i is:

$$\Psi_i = Ue^{-s\Lambda}U^T\delta_i \tag{1}$$

$\delta_i = 1(i)$ is the one-hot vector for node i, Ψ_i is regarded as a set of random variables, based on (1), its characteristic function can be calculated as follows:

$$\phi_i(t) = \frac{1}{N}\sum_{m=1}^{N} e^{it\Psi_{mi}} \tag{2}$$

The role representations of i at time t are:

$$X_r = [Re(\phi_i(t_i)), Im(\phi_i(t_i))]_{t_1,\ldots,t_d} \in \mathcal{R}^{2d} \tag{3}$$

where Re denotes the real parts and Im denotes the imaginary parts.

2) Adjacency Embedding: The adjacency embeddings of the nodes are calculated by Node2vec. Given the current node v, the probability of visiting the next node x is:

$$P(c_i = x|c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{\mathcal{Z}} & \text{if } (v,x) \in E \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$\pi_{vx} = \alpha_{pq}(t,x)\cdot w_{vx}$ is the transition probability between node v and x before normalization, where w_{vx} is the edge weight between node v and x. $\mathcal{Z}$ is the normalization constant. Two hyperparameters p and q are used to calculate the transition probability:

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases} \tag{5}$$

d_{tx} represents the shortest path between node t and node x. The node sequences obtained by random walk are learned by the Skip-Gram[33] to generate the adjacency embedding X_a.

IV. Proposed Method

The overall framework of RAHG is shown in Fig. 2. The workflow is divided into three parts. In the first part, we encode the graph structure and get the initial role and adjacency representations by Graphwave [19] and Node2vec [32], respectively. We also construct a hypergraph from the input graph. Based on these, we obtain a fusion embedding via an attention mechanism. Then the fusion embeddings and hypergraph adjacency matrix are put into the multi-layer hypergraph convolutional networks for iteration computations. We classify the nodes with the trained node representations in the last part.

A. Hypergraph Construction

In this section, we describe two ways of constructing hypergraphs in detail. The constructed hypergraphs are put into the subsequent hypergraph convolution networks for feature aggregations.

1) Hypergraph Description: An edge in typical simple graphs connects two nodes, while a hyperedge in hypergraphs can connect multiple nodes with similar properties as a hyperedge. We can reorganize a simple graph into a hypergraph. Given a hypergraph $G_H = (V_H, E_H, W_H)$, H is written as $h(v,e)$, $h(v,e)$ equals 0 or 1, which means whether node v belongs to a hyperedge e. V_H is the node set, E_H is the hyperedge set. W_H is the hyperedge weights matrix. The degree of each node is

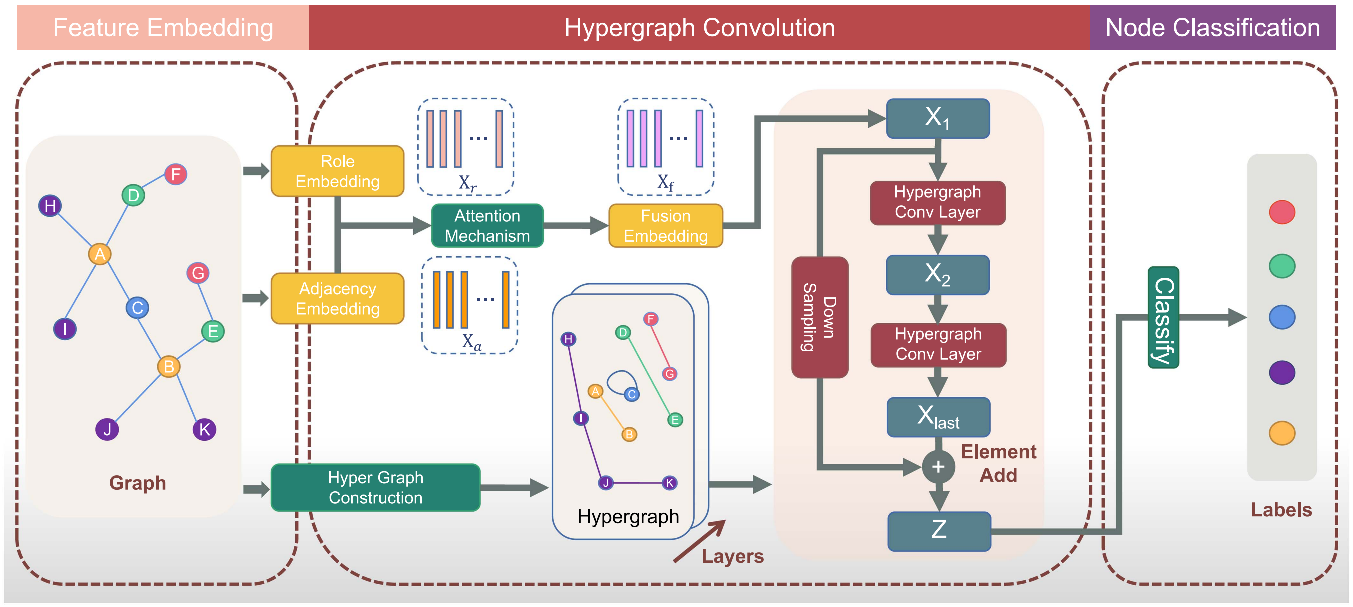

Fig. 2. Framework of the proposed RAHG.

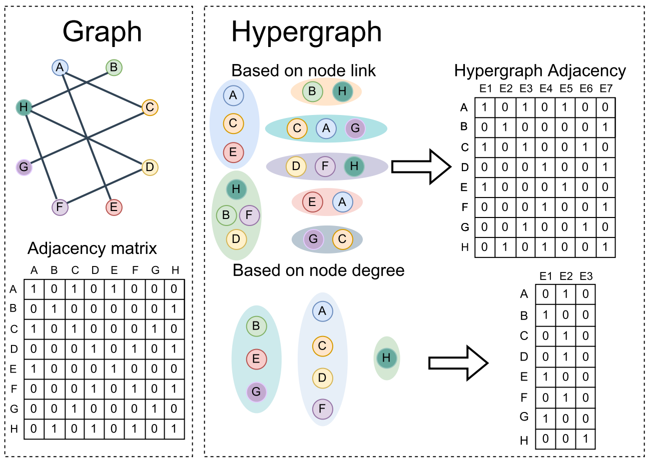

Fig. 3. Examples of hypergraph construction.

expressed as:

$$d(v_i) = \sum_{e_i \in E} h(v_i, e_i) w(e_i) \tag{6}$$

the degree of each hyperedge is written as:

$$d(e_i) = \sum_{v_i \in V} h(v_i, e_i) \tag{7}$$

D_v and D_e are used to represent the degree matrix of nodes and hyperedges in a hypergraph.

The hypergraph constructions of RAHG are shown in Fig. 3.

2) Hypergraph by Node Link: This method constructs a hypergraph G_h^l based on neighbor links of a graph G. G_h^l connects all neighbors of the central nodes to form a hyperedge. The hypergraph structure $H_l \in \mathcal{R}^{(N \times N_E^l)}$ is generated eventually, where N is the number of nodes in G and N_E^l is the number of hyperedges. D_v^l and D_e^l represent the node and edge degree of H_l.

3) Hypergraph by Node Degree: A hypergraph G_h^d is constructed by node degree. A set of nodes with the same degree is classified as a hyperedge. The structure of the generated hypergraph is $H_d \in \mathcal{R}^{(N \times N_E^d)}$, where N is the number of nodes and N_E^d is the number of different degrees of nodes. D_v^d and D_e^d represent the node and the edge degree of H_d.

B. Role-Aware Hypergraph Neural Network

Based on the constructed hypergraphs, we propose a role-aware hypergraph neural network. The pseudocode of RAHG is given in Algorithm 1.

1) Attention Mechanism: RAHG introduces an attention mechanism to make the neural network adaptively cooperate with two types of representations, which benefits downstream tasks. Specifically, the role features are embedded to $\hat{X}_r$ by (8). The attention weight α_r is as follows:

$$\hat{X}_r = X_r W_r + b_r \tag{8}$$

$$\alpha_r = \tanh \hat{X}_r q \tag{9}$$

$$\hat{\alpha_r} = \frac{exp(\alpha_r)}{exp(\alpha_r) + exp(\alpha_a)} \tag{10}$$

$$X_f = \hat{\alpha_r} \hat{X}_r + \hat{\alpha_a} \hat{X}_a \tag{11}$$

where $W_r \in \mathcal{R}^{F_r \times F_h}$ and $b_r \in \mathcal{R}^{F_h}$ are the weight matrix and bias. $q \in \mathcal{R}^{F_h}$ is the attention vector, F_h is the hidden size of layers. $\tanh$ is chosen as the activation function to prevent the gradient from disappearing and improve the convergence speed. The attention weights $\alpha_r \in \mathcal{R}^N$ of role embedding can be calculated by (9). Then in (10), $softmax$ is used to normalize attention weights of the role and adjacency embeddings. The final fusion embedding $X_f \in \mathcal{R}^{N \times F_h}$ is calculated by (11) and used as the input features of the hypergraph convolutional networks.

Algorithm 1: Framework of RAHG.

```
   Data: general graph $G = (V, E, A)$.
   Result: the final node embedding $Z$.
1  Initialize the parameters of model $\theta$ randomly;
2  Construct hypergraph $G_H^l$ or $G_H^d$ with Adjacency $A$;
3  Initialize the role embedding $X_r$ and adjacency
     embedding $X_a$, respectively;
4  while not converged do
5      Obtain the fusion embedding by attention
         mechanism (Eqs. (8) (9) (10) (11));
6      Obtain the $X_{last}$ by multi hypergraph convolution
         layers (Eq. (12));
7      Obtain the final embedding $Z$ by residual network
         (Eq. (13));
8      Update $\theta$ by cross-entropy loss (Eqs. (14) (15));
9  end
10 Return the final node embedding $Z$;
```

2) Hypergraph Convolution Layer: The hypergraph convolutional network takes the fusion features generated by the attention mechanism based on (11) as the initial input. The convolution formula of node fusion features X based on HGNN is as follows:

$$X_l = \sigma\left(D_v^{-1} H W_H D_e^{-1} H^T D_v^{-1} X_{l-1} \theta_{l-1}\right) \tag{12}$$

where W_H is the weight matrix of hyperedges, D_v and D_e are the degree matrices of nodes and hyperedges, H is the hypergraph structure, X_{l-1} is the input of lth layer, X_l is the output of lth layer, θ_{l-1} is the parameters of lth layer, σ is the nonlinear activation function.

3) Residual Network: To alleviate the over-smoothing problem of GNNs, we map the raw representation by down-sampling and add raw features to the output embedding of the last HGNN layer. The calculation formula is as follows:

$$Z = X_{last} + X_1 W_{res} \tag{13}$$

where $W_{res} \in \mathcal{R}^{F_h \times F_h}$ and $X_1 \in \mathcal{R}^{N \times F_h}$ are weight matrix and the raw representation, respectively. $X_{last} \in \mathcal{R}^{N \times F_h}$ is the outputs of the last layer, $Z \in \mathcal{R}^{N \times F_h}$ is the final representation after the residual network.

C. Classification

Z is fed into a dense network with $softmax$ to predict value $\hat{Y}$:

$$\hat{Y} = softmax(ZW_m + b_m) \tag{14}$$

where $W_m \in \mathcal{R}^{F_h \times M}$ and $b_m \in \mathcal{R}^M$ are weight matrix and bias, respectively. The loss function is:

$$\mathcal{L} = -\sum_{v \in V} \sum_{i=1}^{M} Y \ln \hat{Y} + \lambda \left\|\theta\right\|^2 \tag{15}$$

where λ denotes a regularization parameter and θ denotes the parameters of the model.

V. Experiments

In this section, extensive experiments on datasets of different scales and approaches are conducted to verify the performance of the RAHG.

A. Experiments Settings

1) Datasets: The model's performance is tested in a new dataset and six public datasets. Statistics of the datasets, including heterogeneity and assortative coefficients [45], are summarized in Table II. Visualizations of datasets are shown in Fig. 4. The details are as follows:

- *ENZYMES:* The datasets of 600 tertiary protein structures were obtained from the BRENDA enzymes database [46]. We selected the enzymes with more than 90 nodes, with the indexes 118, 123, 295, 295, 296, and 297. Each enzyme graph has two types of nodes. We use E118 to represent ENZYMES118 in short. The other ENZYMES datasets are represented in the same way.
- *Internet-industry-partnerships (IIP) [47]:* The nodes in the network represent companies competing in the internet industry. If two companies have announced joint ventures, strategic alliances, or other partnerships, they are connected by an edge. The nodes' three tags include Content, Infrastructure, and Commerce.
- *TerroristRel:* A public dataset collected from the PIT repository [48]. This dataset contains information about terrorists and their relationships. A vector of 0/1 values describes each relationship.
- *Cora [49]:* It includes 2708 scientific publications on machine learning, nodes are divided into seven categories.
- *Citeseer [49]:* It includes 3327 scientific publications, and nodes are divided into six categories.
- *Pubmed [49]:* It includes 19717 scientific publications on diabetes from the Pubmed database, and nodes are divided into three categories.
- *Adv-Stu:* The Adv-Stu dataset is a real-world network dataset constructed in this paper. There are 44 graduate students and 26 advisors (professors) at Anhui University, Hefei, China. If an advisor instructs a student, there is a connection between them. If two advisors are in the same research group, they also have a connection. The nodes in the graph are labeled as advisors and students. The structure of the graphs is represented in Fig. 4(f), index from 0 to 43 indicating students and 44 to 69 indicating advisors. Colors of nodes present roles.

The nodes composed of ENZYMES are divided into primary and secondary ones as described in [46]. The IIP has a vast market relationship between big companies and startups [47]. There are differences in the status of factions in TerroristRel [48]. In Cora, Citeseer, and Pubmed, the core papers are widely cited by other papers [49]. Adv-Stu also contains two kinds of role nodes: supervisor and graduates. The above datasets from different domains and sizes can effectively reflect the adjacency and role characteristics of nodes, which are suitable for the verification of the effectiveness of models.

TABLE II
DATASETS

Dataset	E118	E123	E295	E296	E297	Adv-Stu	IIP	TerroristRel	Cora	Citeseer	Pubmed
$\|V\|$	95	90	123	125	121	70	219	881	2708	3327	19717
$\|E\|$	121	127	139	141	149	72	631	8592	10556	9104	88648
# classes	2	2	2	2	2	2	3	2	7	6	3
#Avg degree	2.55	2.82	2.26	2.26	2.46	2.06	2.91	9.75	3.90	2.78	4.50
#Avg k-Core	3.75	3.91	3.24	3.71	3.92	2.14	3.18	17.60	4.63	3.52	4.79
#Heterogeneity coefficient	0.17	0.23	0.15	0.13	0.17	0.40	0.53	0.274	0.41	0.44	0.60
#Assortative coefficient	0.10	0.16	0.21	0.29	0.16	-0.25	-0.12	-0.02	-0.07	0.05	-0.04
#Imbalanced ratio	1:0.79	1:0.40	1:0.18	1:0.23	1:0.19	1:0.59	1:0.33:0.32	1:0.27	-	-	1:0.98:0.52

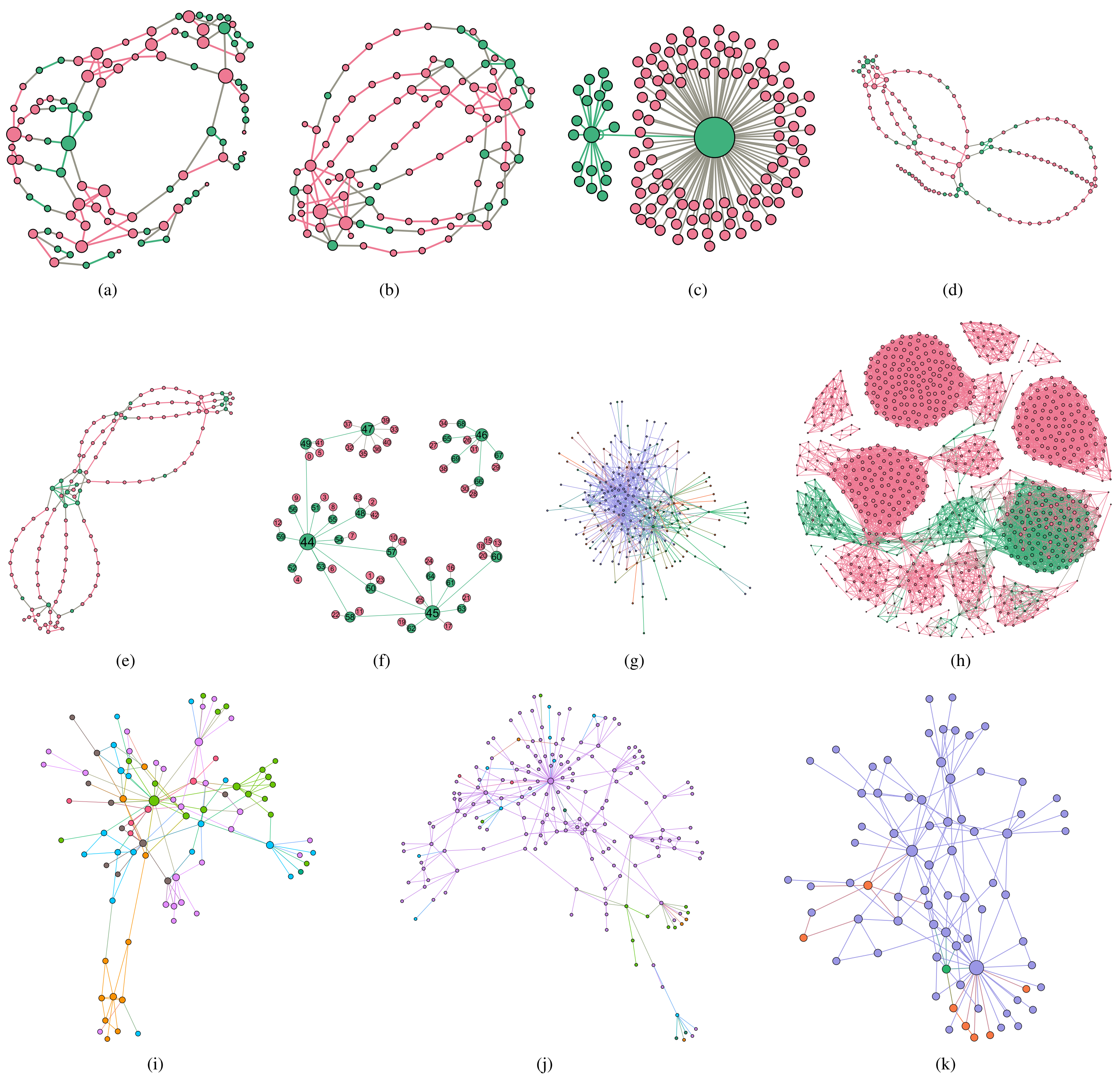

Fig. 4. Visualization of datasets. (a) E118. (b) E123. (c) E295. (d) E296. (e) E297. (f) Adv-Stu. (g) IIP. (h) TerroristRel. (i) A subgraph of Cora. (j) A subgraph of Citeseer. (k) A subgraph of Pubmed.

TABLE III
AVERAGE PERFORMANCE ON NODE CLASSIFICATION

Method	E118	E123	E295	E296	E297	Adv-Stu	IIP	TerroristRel	Cora	Citeseer	Pubmed
Role	54.55±9.89	46.23±13.15	45.21±11.80	42.87±9.46	44.41±12.28	62.40±8.64	60.62	52.11	30.23	19.62	39.05
Adj	53.66±9.99	47.23±13.37	44.86±12.39	42.02±10.44	47.31±12.39	63.62±8.77	60.79	51.91	30.02	19.40	71.65
Role+Adj	52.07±8.44	70.54±13.07	45.91±13.50	41.15±9.67	45.37±11.10	59.79±11.68	59.50	51.33	30.02	19.35	39.77
Deepwalk	54.76±8.38	46.35±12.44	45.73±12.69	42.66±9.05	45.70±11.99	64.19±8.91	60.85	51.77	30.09	33.60	33.20
Struc2vec	55.38±10.02	46.01±13.11	45.16±12.81	42.65±9.47	46.35±10.60	63.62±9.37	61.24	51.91	30.06	-	-
Role+GCN	58.55±9.44	61.74±14.11	61.93±12.89	59.74±13.10	59.56±12.26	63.76±9.49	60.23	55.17	38.74	39.38	38.53
Adj+GCN	58.72±8.33	61.23±14.07	61.67±12.73	59.32±12.94	61.04±12.31	57.95±9.39	58.73	55.92	71.74	51.84	70.72
Role+GAT	54.72±9.58	52.53±14.79	58.55±15.59	54.94±13.78	58.23±13.80	64.05±9.70	60.18	53.24	40.51	41.24	35.70
Adj+GAT	56.55±9.86	56.89±14.17	59.31±14.91	56.23±12.59	60.09±13.84	64.86±9.95	60.08	53.25	71.96	51.82	71.90
Role+HGCN(d)	54.31±9.63	52.68±14.59	50.89±13.47	49.20±14.10	50.89±14.14	62.81±9.40	60.41	53.42	38.74	24.70	28.54
Role+HGCN(l)	57.10±9.88	54.10±15.51	58.14±14.52	56.88±12.82	58.24±12.80	57.24±13.90	60.18	53.73	44.16	42.39	41.21
Adj+HGCN(d)	55.28±9.58	54.51±14.53	51.55±13.34	51.05±13.68	51.55±14.39	55.28±9.40	58.95	53.78	30.06	24.89	35.20
Adj+HGCN(l)	58.07±8.63	57.10±14.61	58.48±13.79	57.97±13.13	58.94±13.24	57.48±14.13	58.64	53.96	72.26	51.74	73.17
RAHG(d)	62.03±6.81	68.96±9.05	68.77±9.54	67.45±8.86	70.00±10.24	**67.50**±12.61	61.50	60.99	71.56	51.39	74.86
RAHG(l)	**64.66**±6.99	**71.29**±10.28	**73.20**±10.97	**71.46**±8.94	**73.18**±10.45	67.37±11.28	**61.77**	**61.31**	**73.14**	**52.46**	**75.18**
Absolute improvement	5.94	0.75	11.27	11.72	12.14	2.64	0.53	5.39	0.88	0.62	2.01
Improvement ratio	10.12%	1.06%	18.20%	19.62%	19.89%	4.07%	0.87%	9.64%	1.22%	1.20%	2.75%

The bold values highlighted the best performance among the methods.

2) Baselines: We compare the RAHG with the following strong baselines:

- *Role:* Role representations of nodes by GraphWave [19] are used as input features of SVM[50] with RBF (Radial Basis Function) for the node classification task.
- *Adj:* Adjacency characteristics of nodes by Node2vec [32] are used as input features of the classifier by the SVM.
- *Role+Adj:* Role and adjacency features are concatenated as the input features of the classifier by the SVM.
- *Deepwalk:* Deepwalk [31] is a graph embedding method that produces node representations by randomly walking and Skip-Gram [33] in a graph. We apply the same classifier.
- *Struc2vec:* Struc2vec [38] defines vertex similarity from a role-based biased Markov walks.
- *Role+GCN:* The role embeddings by GraphWave [19] as the input features of GCN for classification.
- *Adj+GCN:* The adjacency embeddings by Node2vec [32] as the input features of GCN for classification.
- *Role+GAT:* The role embeddings by GraphWave [19] as the input features of GAT for classification.
- *Adj+GAT:* The adjacency embeddings by Node2vec [32] as the input features of GAT for classification.
- *Role+HGCN:* The role embeddings by GraphWave [19] as the input features for the hypergraph convolutional network [20] for classification.
- *Adj+HGCN:* The adjacency embeddings by Node2vec [32] as the input features the same hypergraph convolutional network for classification.

3) Parameter Settings: For the RAHG, we apply two hypergraph convolution layers and one fully connected layer. p and q in the Node2vec are both set to 1.0. F_r and F_a are both set to 128. Adam is applied as the optimizer. The initial learning rate is set to 0.003. The coefficient of dropout is chosen from $\{0.3, 0.4, 0.5, 0.6, 0.7\}$. The hidden size F_h is set from $\{60, 80, 100, 120, 140\}$. The weight decay coefficient is set to 5e-4. For the compared baselines, we set the parameters to their default values.

TABLE IV
MAXIMUM PERFORMANCE ON NODE CLASSIFICATION

Method	IIP	TerroristRel	Cora	Citeseer	Pubmed
Role	72.73	67.57	32.60	23.10	39.10
Adj	72.73	67.57	32.90	22.90	72.50
Role+Adj	72.73	70.00	32.10	22.80	43.80
Deepwalk	72.73	71.11	33.60	23.40	33.80
Struc2vec	74.24	70.59	33.00	-	-
Role+GCN	72.73	71.43	41.40	39.90	40.80
Adj+GCN	70.45	69.39	73.50	52.70	70.90
Role+GAT	72.73	65.85	42.80	42.90	41.80
Adj+GAT	72.73	65.85	73.40	53.30	72.40
Role+HGCN(d)	75.00	65.85	41.31	25.20	40.70
Role+HGCN(l)	72.73	65.85	48.10	43.40	42.10
Adj+HGCN(d)	70.45	65.85	32.90	25.80	37.10
Adj+HGCN(l)	70.45	65.85	73.10	53.90	73.70
RAHG(d)	75.00	79.07	72.70	**54.66**	75.60
RAHG(l)	**79.55**	**79.41**	**75.80**	54.15	**77.80**
Absolute improvement	4.55	7.98	2.3	0.76	4.10
Improvement ratio	6.07%	11.17%	3.13%	1.41%	5.56%

The bold values highlighted the best performance among the methods.

B. Node Classification

The node classification task is used to verify the performance of RAHG and baselines. Table II shows that the ENZYMES and TerroristRel have imbalanced labels. Uniform partition of

TABLE V
ABLATION STUDY ON NODE CLASSIFICATION

Dataset	E118	E123	E295	E296	E297	Adv-Stu	IIP	TerroristRel	Cora	Citeseer	Pubmed
$RAHG(d)$	62.03	68.96	68.77	67.45	70.00	**67.50**	61.50	61.05	71.56	51.39	74.86
$RAHG(d)-\{residual\}$	59.38	64.48	64.04	63.09	63.89	63.57	59.86	60.07	32.27	25.42	43.90
$RAHG(d)-\{attention\}$	61.31	66.93	66.06	65.46	64.89	65.14	60.45	60.09	70.90	50.35	73.61
$RAHG(l)$	**64.66**	**71.29**	**73.20**	**71.46**	**73.18**	67.37	**61.77**	**61.31**	**73.14**	**52.46**	**75.18**
$RAHG(l)-\{residual\}$	63.79	69.39	70.85	69.69	69.41	64.14	61.27	60.26	72.32	50.93	74.20
$RAHG(l)-\{attention\}$	63.28	69.20	70.39	68.07	68.57	64.93	60.91	58.62	71.89	51.63	74.78

The bold values highlighted the best performance among the methods.

these datasets will make the model more focused on classes with more data samples. We divided the train and test sets by random sampling for imbalanced labels between categories in imbalanced datasets. Specifically, the larger node set is randomly sampled to keep the ratio to the smaller node set within 1:0.33.

Note that the original features in Cora, Citeseer, and Pubmed are not utilized for node classification (only structural features of graphs are applied), so the performance of some baselines is different from the original papers [10]. Due to the small scale of the graph structure of ENZYMES and Adv-Stu, the prediction accuracy can be serendipitous. To make experiments convincing, we report models' average and maximum performance by running models 100 times on the ENZYMES and Adv-Stu datasets, 50 times on IIP and TerroristRel datasets, and ten experiments on Cora, Citeseer, and Pubmed under the same data preprocessing. The graph structure of IIP, TerroristsRel, Cora, Citeseer and Pubmed is large, and the best performances on these datasets are expected. The average micro-F1 and the best Micro-F1 are presented.

The results are listed in Tables III and IV. We use Micro-F1 as the indicator in the node classification task. Micro-F1 is a common-used metric in evaluating classification performance and can reflect the effectiveness of models [10], [11], [14]. The Micro-F1 $= 2(Recall_{micro} \times Precision_{micro})/(Recall_{micro} + Precision_{micro})$. The Absolute improvement $= M_1 - M_2$, the Improvement ratio $= (M_1 - M_2)/M_2 \times 100\%$, where M_1 is the best performance of RAHG and M_2 is the best performance of the baselines. RAHG achieves the best performance on all datasets and outperforms all the other comparative models on the other datasets by large margins. The neighbor links of the node better reflect the local subgraph structure of the node in ENZYMES, IIP, TerroristRel, Cora, Citeseer, and Pubmed. Moreover, the labels of nodes with more complex neighbor adjacencies are easier to distinguish. Thus, $RAHG(l)$ outperforms $RAHG(d)$ in these datasets.

In Adv-Stu, node classification pays more attention to the number of nodes connected, and the performance of $RAHG(d)$ and $RAHG(l)$ is almost the same. Both $RAHG(d)$ and $RAHG(l)$ perform much better than strong baselines in terms of structural role representation in hypergraph neural networks.

C. Ablation Study

We explore the contributions of each component of RAHG in ablation studies. The results are listed in Table V. RAHG has two variant models based on node degree and node link denoted as $RAHG(d)$ and $RAHG(l)$, respectively. $-\{residual\}$ and $-\{attention\}$ represent the removal of the residual network and attention mechanism in RAHG, respectively. Removing residual network and attention mechanism both lead to a reduction of accuracy by large margins. These two mechanisms both play significant roles in RAHG. It is worth noting that $RAHG(d)$ is severely regressed in three large citation network datasets after removing the residual network. The residual networks add the initial representations with global network information by the graph representation learning methods, which benefits the global optimization based on $RAHG(d)$. However, the improvement

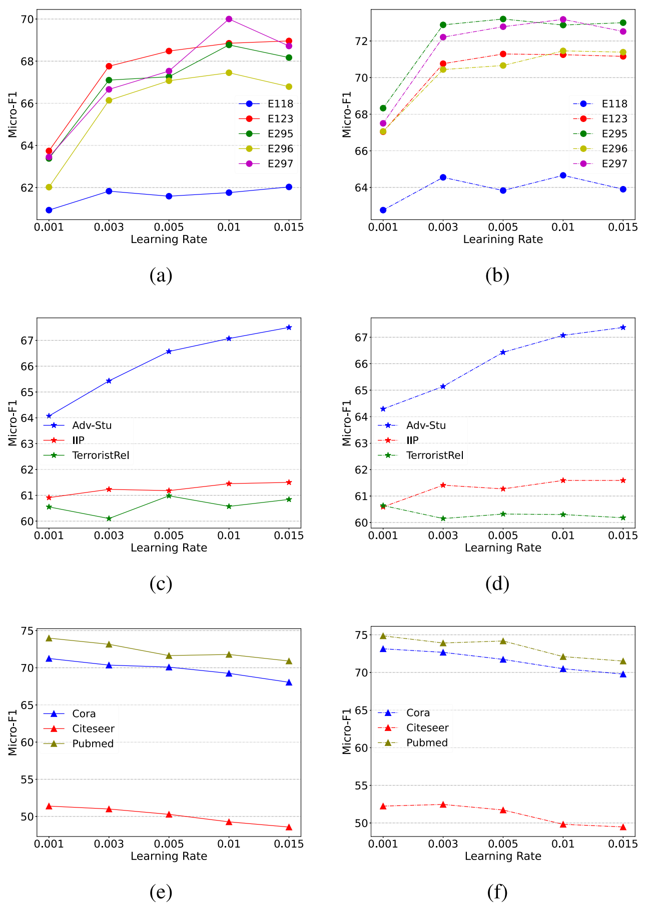

Fig. 5. How the learning rate impacts the performance. (a) ENZYMES*(RAHG(d))*. (b) ENZYMES*(RAHG(l))*. (c) AS, IIP and TR*(RAHG(d))*. (d) AS, IIP and TR*(RAHG(l))*. (e) CR, CS and Pb*(RAHG(d))*. (f) CR, CS and Pb*(RAHG(l))*.

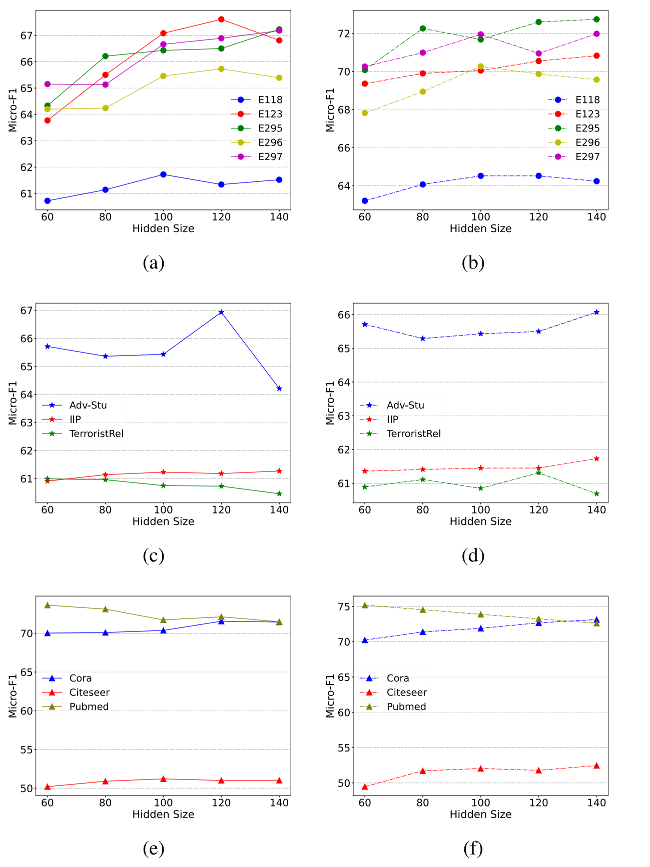

Fig. 6. How the hidden size impacts the performance. (a) ENZYMES*(RAHG(d))*. (b) ENZYMES*(RAHG(l))*. (c) AS, IIP and TR*(RAHG(d))*. (d) AS, IIP and TR*(RAHG(l))*. (e) CR, CS and Pb*(RAHG(d))*. (f) CR, CS and Pb*(RAHG(l))*.

by $RAHG(l)$ is not obvious. The residual network is more critical for $RAHG(d)$ than for $RAHG(l)$. In contrast, the attention mechanism plays a more significant role in $RAHG(l)$.

D. Parameters Sensitivity

AS, TR, CR, CS, and Pb denote Adv-Stu, TerroristRel, Cora, Citeseer, and Pubmed in short, respectively. From Fig. 5, models have different sensitivities with learning rates. RAHG on ENZYMES, Adv-Stu, and IIP achieve better performance with a relatively larger learning rate. A lower learning rate is recommended on Cora, Citeseer, and Pubmed.

The parameter sensitivities of $RAHG(l)$ and $RAHG(d)$ are generally similar. $RAHG(l)$ is less sensitive to the hidden size F_h than $RAHG(d)$ as shown in Fig. 6 on ENZYMES. The performances are increased with larger hidden sizes from 60 to 140. A hidden size of 120 or 140 is recommended. For citation networks, Cora and Citeseer are suitable for the larger hidden size, while Pubmed performs better in smaller hidden sizes.

E. Visualization

The node-level representations are processed in two dimensions by TSNE [51] to explore the differences between RAHG and baselines. As shown in Fig. 7, the distributions of nodes before training are both chaotic. After training, the nodes are more clustered according to their class labels through both RAHG and Deepwalk. However, the nodes are more densely gathered in RAHG.

The attention weights of RAHG after training in each dataset are shown in Fig. 8. RAHG assigns higher weights to the adjacency features on E118, E123, E296, E297, IIP, Cora, Citeseer, and Pubmed. From Table II and Fig. 4, RAHG tends to assign larger weights on adjacency representations in smaller and sparser graphs. On the contrary, the model pays more attention to the role features of nodes on E295, TerroristRel, and Adv-Stu, which indicates that structural role might have more

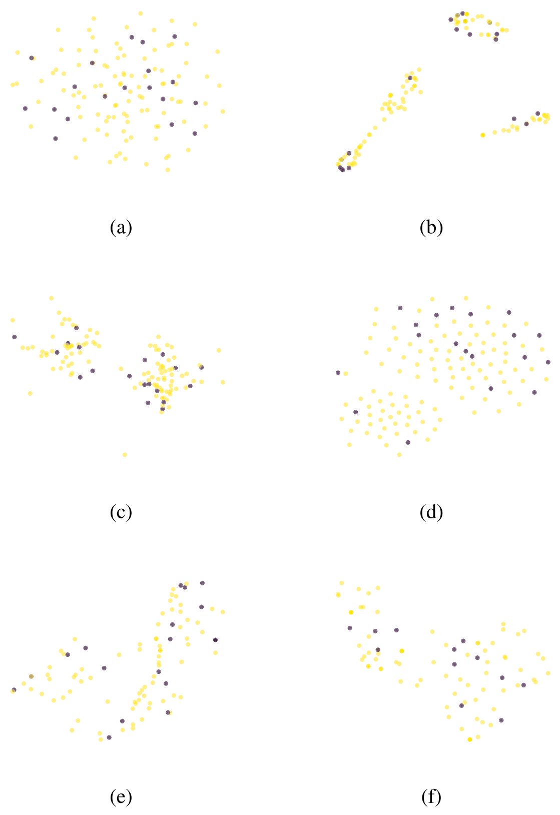

Fig. 7. Visualization of node classification task on ENZYMES295. (a) Before training. (b) RAHG. (c) Deepwalk. (d) Struc2vec. (e) Role+GCN. (f) Adj+GCN.

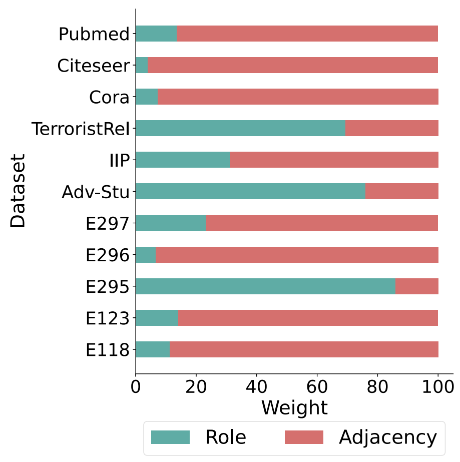

Fig. 8. Attention weights.

influence in higher-density graphs or datasets with higher-degree nodes.

VI. Conclusion

Many nodes with similar roles are not always directly connected in specific graph datasets. Traditional graph neural networks fail to perform well in these datasets without considering role representations. We propose RAHG, a novel method aggregating node role and adjacency representations by an attention mechanism and hypergraph neural networks. Extensive experiments demonstrate that the RAHG achieved SOTA performances and outperforms baselines by large margins, which provides a new perspective for designing new graph neural networks. In addition, RAHG is scalable and can be used as a general graph representation framework to apply to any networks with role characteristics.

However, many graphs are represented in heterogeneous graphs. Nodes' role and adjacency attributes have more complex aggregations and restrictions on heterogeneous graphs. RAHG cannot currently handle the association of heterogeneous roles. In the future, we will extend RAHG to more types of graphs and tasks.

Acknowledgment

The authors also thank PyG [52] for providing the framework for our experiments.

References

[1] S. Min, Z. Gao, J. Peng, L. Wang, K. Qin, and B. Fang, "STGSN–A spatial–temporal graph neural network framework for time-evolving social networks," *Knowl.-Based Syst.*, vol. 214, 2021, Art. no. 106746.
[2] S. Dhelim, N. Aung, and H. Ning, "Mining user interest based on personality-aware hybrid filtering in social networks," *Knowl.-Based Syst.*, vol. 206, 2020, Art. no. 106227.
[3] M. Tan, W. Chen, W. Wang, A. Liu, and L. Zhao, "Intention-oriented hierarchical bundle recommendation with preference transfer," in *Proc. IEEE Int. Conf. Web Serv.*, 2021, pp. 107–116.
[4] Z. Li et al., "Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications," in *Proc. IEEE 36th Int. Conf. Data Eng.*, 2020, pp. 1677–1688.
[5] Y. Wu, H.-N. Dai, and H. Tang, "Graph neural networks for anomaly detection in industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9214–9231, Jun. 2022.
[6] C. Q. Nguyen, C. Kreatsoulas, and K. M. Branson, "Meta-learning GNN initializations for low-resource molecular property prediction," in *Proc. 4th Lifelong Mach. Learn. Workshop ICML*, 2020.
[7] K. Huang, C. Xiao, L. M. Glass, M. Zitnik, and J. Sun, "SkipGNN: Predicting molecular interactions with skip-graph networks," *Sci. Rep.*, vol. 10, no. 1, 2020, Art. no. 21092.
[8] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.
[9] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 249–270, Jan. 2022.
[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.
[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018.
[12] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 1025–1035.
[13] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*.
[14] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proc. Int. Conf. Learn. Representations*, 2019.
[15] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*2017, pp. 1263–1272.
[16] Y. Yang et al., "Rain: Social role-aware information diffusion," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015.
[17] K. Henderson et al., "Rolx: Structural role extraction & mining in large graphs," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2012, pp. 1231–1239.
[18] N. K. Ahmed et al., "role2vec: Role-based network embeddings," *Proc. 8th Int. Workshop Deep Learn. Graphs: Methods Appl.*, 2019, pp. 1–7.
[19] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1320–1329.
[20] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 3558–3565.
[21] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, vol. 29, pp. 3844–3852.
[22] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 639–648.
[23] M. Yang et al., "Hyperbolic graph neural networks: A review of methods and applications," 2022, *arXiv:2202.13852*.
[24] D. Chen, L. O'Bray, and K. Borgwardt, "Structure-aware transformer for graph representation learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 3469–3489.
[25] J. You, J. M. Gomes-Selman, R. Ying, and J. Leskovec, "Identity-aware graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, pp. 10737–10745.
[26] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7134–7143.
[27] X. Li et al., "Finding global homophily in graph neural networks when meeting heterophily," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2022, pp. 13 242–13 256.
[28] G.-S. Xie, J. Liu, H. Xiong, and L. Shao, "Scale-aware graph neural network for few-shot semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 5475–5484.
[29] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, "Graph wavelet neural network," in *Proc. Int. Conf. Learn. Representations*, 2019.
[30] F. Chen, Y.-C. Wang, B. Wang, and C.-C. J. Kuo, "Graph representation learning: A survey," *APSIPA Trans. Signal Inf. Process.*, vol. 9, 2020, Art. no. e15.
[31] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.
[32] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.
[33] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.
[34] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.
[35] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1225–1234.
[36] C. Ying et al., "Do transformers really perform badly for graph representation?," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 28877–28888.
[37] Z. Wang, Q. Cao, H. Shen, B. Xu, K. Cen, and X. Cheng, "Location-aware convolutional neural networks for graph classification," *Neural Netw.*, vol. 155, pp. 74–832022.
[38] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 385–394.
[39] D. Zhou and J. Huang, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, B. Schölkopf, J. C. Platt, and T. Hofmann Eds. 2007, pp. 1601–1608.
[40] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar, "HyperGCN: A new method for training graph convolutional networks on hypergraphs," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1511–1522.

[41] S. Bai, F. Zhang, and P. H. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognit.*, vol. 110, 2021, Art. no. 107637.
[42] R. Zhang, Y. Zou, and J. Ma, "Hyper-SAGNN: A self-attention based graph neural network for hypergraphs," in *Proc. Int. Conf. Learn. Representations*, 2020.
[43] E. Chien, C. Pan, J. Peng, and O. Milenkovic, "You are allset: A multiset function framework for hypergraph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2022.
[44] T. Yang et al., "Co-clustering interactions via attentive hypergraph neural network," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2022, pp. 859–869.
[45] N. Samadi and A. Bouyer, "Identifying influential spreaders based on edge ratio and neighborhood diversity measures in complex networks," *Computing*, vol. 101, no. 8, pp. 1147–1175, 2019.
[46] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl_1, pp. i47–i56, 2005.
[47] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proc. AAAI Conf. Artif. Intell.*, 2015. [Online]. Available: https://networkrepository.com
[48] B. Zhao, P. Sen, and L. Getoor, "Event classification and relationship labeling in affiliation networks," in *Proc. Workshop Stat. Netw. Anal., 23rd Int. Conf. Mach. Learn.*, 2006, pp. 271–280.
[49] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," 2016, *arXiv:1603.08861*.
[50] W. S. Noble, "What is a support vector machine?," *Nature Biotechnol.*, vol. 24, no. 12, pp. 1565–1567, 2006.
[51] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008.
[52] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *Proc. Int. Conf. Learn. Representations Workshop Representation Learn. Graphs Manifolds*, 2019.

Kunhao Li is currently working toward the graduation degree with Anhui University, Hefei, China, (Project 211, Double First-Class Initiative). His research focuses on graph neural networks.

Zhenhua Huang was a joint Ph.D. from the South China University of Technology and the University of California, Irvine. He is currently an assistant professor with Anhui University, Hefei, China. His research interests include complex networks, network embedding, and graph neural networks. He has authored or coauthored papers in AAAI, WWW, Physica A, IEEE TRANSACTIONS ON HUMAN-MACHINE SYSTEMS, and *Science China Information Sciences*.

Zhaohong Jia (Member, IEEE) received the Ph.D. degree from the University of Science and Technology of China, Hefei, China. She is currently a Professor from Anhui University, Hefei, and She has authored or coauthored papers in IEEE TRANSACTIONS ON CLOUD COMPUTING, *Expert Systems with Applications*, *Applied Soft Computing*, *Applied Intelligence,* and *Future Generation Computer Systems*. Her research interests include evolutionary algorithms, data mining, and pattern recognition.