MHGCN+: Multiplex Heterogeneous Graph Convolutional Network

CHAOFAN FU*, Ocean University of China, Qingdao, China
PENGYANG YU*, Ocean University of China, Qingdao, China
YANWEI YU†, Ocean University of China, Qingdao, China
CHAO HUANG, The University of Hong Kong, Hong Kong, China
ZHONGYING ZHAO, Shandong University of Science and Technology, Qingdao, China
JUNYU DONG, Ocean University of China, Qingdao, China

Heterogeneous graph convolutional networks have gained great popularity in tackling various network analytical tasks on heterogeneous graph data, ranging from link prediction to node classification. However, most existing works ignore the relation heterogeneity with multiplex networks between multi-typed nodes and the different importance of relations in meta-paths for node embedding, which can hardly capture the heterogeneous structure signals across different relations. To tackle this challenge, this work proposes a **M**ultiplex **H**eterogeneous **G**raph **C**onvolutional **N**etwork (MHGCN+) for multiplex heterogeneous network embedding. Our MHGCN+ can automatically learn the useful heterogeneous meta-path interactions of different lengths with different importance in multiplex heterogeneous networks through multi-layer convolution aggregation. Additionally, we effectively integrate both multi-relation structural signals and attribute semantics into the learned node embeddings with both unsupervised and semi-supervised learning paradigms. Extensive experiments on seven real-world datasets with various network analytical tasks demonstrate the significant superiority of MHGCN+ against state-of-the-art embedding baselines in terms of all evaluation metrics. The source code of our method is available at: https://github.com/FuChF/MHGCN-plus.

CCS Concepts: • **Mathematics of computing → Graph algorithms**; • **Computing methodologies → Learning latent representations**.

Additional Key Words and Phrases: Network Embedding, Graph Representation Learning, Multiplex Heterogeneous Networks, Graph Convolutional Networks

1 INTRODUCTION

Network representation learning is emerging as a new paradigm of learning to embed a complex network in a lower-dimensional space, while preserving the proximities of the nodes, both in the topological structure and in the intrinsic properties of the network. Effective network representation facilitates various network analytical tasks, ranging from link prediction [4, 20, 26], node classification [8, 21, 22, 39], to recommendation [14, 15, 30, 44, 51]. In recent years, Graph Convolutional Networks (GCNs) [17], a class of neural networks designed to learn graph

*Chaofan Fu and Pengyang Yu contribute equally to this work.
†Yanwei Yu is the corresponding author.

Authors' addresses: Chaofan Fu, Ocean University of China, Qingdao, China, fuchaofan@stu.ouc.edu.cn; Pengyang Yu, Ocean University of China, Qingdao, China, 266100, ypy@stu.ouc.edu.cn; Yanwei Yu, Ocean University of China, Qingdao, China, yuyanwei@ouc.edu.cn; Chao Huang, The University of Hong Kong, Hong Kong, China, chaohuang75@gmail.com; Zhongying Zhao, Shandong University of Science and Technology, Qingdao, China, zyzhao@sdust.edu.cn; Junyu Dong, Ocean University of China, Qingdao, China, dongjunyu@ouc.edu.cn.

ACM 2157-6904/2024/2-ART
https://doi.org/10.1145/3650046

representation for complex networks with rich feature information, have been successfully applied to many online services, such as E-commerce [42], social media platforms [40] and advertising [10].

While many efforts have been made to study the representation learning over homogeneous graphs [8, 17, 27, 32, 47], the exploration of preserving network heterogeneous properties in graph representation paradigms has attracted much attention in recent studies, *e.g.*, metapath2vec [5], HIN2Vec [6], and HERec [30]. Inspired by the strength of Graph Neural Networks (GNNs) in aggregating contextual signals from neighboring nodes, various graph neural network models have been introduced to tackle the challenge of heterogeneous graph learning, such as HAN [34], MAGNN [7] and HetGNN [49].

Albeit the effectiveness of existing heterogeneous network embedding approaches [5, 13, 15, 24, 29, 35], these works are generally designed for heterogeneous networks with a single view. In real-world scenarios, however, many networks are much more complex, comprising not only multi-typed nodes and diverse edges even between the same pair-wise nodes but also a rich set of feature attributes [2]. For example, in E-commerce networks, there are two types of nodes (*i.e.*, users and items), and multiple relations (*e.g.*, click, purchase, add-to-cart, or add-to-preference) between the same pairs of users and items [41]. The connections between multiple types of nodes in such networks are often heterogeneous with relation diversity, which yields networks with multiple different views. It is worth noting that the multiplicity of the network is fundamentally different from the heterogeneity of the network. Two types of nodes, users and items, in an E-commerce network, reflect the heterogeneity of the network. At the same time, users may have several types of interactions (*e.g.*, click, purchase, add-to-cart) with items [38], which reflects the multiplex relationships of the network. Because different user-item interactions exhibit different views of the user and item, and thus should be treated differently. We term this kind of networks with multiplex network structures with multi-typed nodes and node attribute information as ***attributed multiplex heterogeneous networks*** (AMHENs).

Performing representation learning on the AMHENs is of great importance to network mining tasks, yet it is very challenging due to such complicated network structures and node attributes. While some recent studies propose to solve the representation learning problem on multiplex heterogeneous network [2, 21, 25, 43, 48], several key limitations exist in those methods: i) The success of current representation learning models largely relies on the accurate design of meta-paths. How to design an automated learning framework to explore the complex meta-path-based dependencies over the multiplex heterogeneous graphs, remains a significant challenge. ii) Unlike the homogeneous node aggregation scheme, with the heterogeneous node types and multiplex node relationships, each meta-path can be regarded as a relational information channel. An effective meta-path dependency encoder is a necessity to inject both the relation heterogeneity and multiplexity into the node representations.

To address the aforementioned challenges, we propose a new **M**ultiplex **H**eterogeneous **G**raph **C**onvolutional **N**etwork, named **MHGCN**, for AMHEN embedding. Particularly, we first decouple the multiplex network into multiple homogeneous and bipartite sub-graphs, and then re-aggregate the sub-graphs with the exploration of their importance (*i.e.*, weights) in node representation learning. To automatically capture meta-path information across multi-relations, we tactfully design a multilayer graph convolution module, which can effectively learn the useful heterogeneous meta-path interactions of different lengths in AMHENs through multilayer convolution aggregation in both unsupervised and semi-supervised learning paradigms. To improve the model efficiency, we endow our MHGCN with a simplified graph convolution for feature aggregation, in order to significantly reduce the model computational cost. Our evaluations are conducted on several real-world graph datasets to evaluate the model performance in both link prediction and node classification tasks. Experimental results show that our MHGCN framework can obtain substantial performance improvement compared with state-of-the-art graph representation techniques.

In this paper, we further extend MHGCN to a more powerful framework, MHGCN+, which explores the more efficient aggregation methods for heterogeneous multi-relations in AMHENs to solve the conflicting issue of the importance of different meta-paths.

We summarize the contributions of this paper as follows:

- We propose an effective multiplex heterogeneous graph neural network, MHGCN+, which can automatically capture the useful relation-aware topological structural signals between nodes for heterogeneous network embedding.
- MHGCN+ integrates both network structures and node attribute features in node representations, and gains the capability to efficiently learn network representation with a simplified convolution-based message passing mechanism.
- We conduct extensive experiments on seven real-world datasets to verify the superiority of our proposed model in both link prediction and node classification when competing with state-of-the-art baselines.

While this work is based on our previous conference article [46], the scope of the proposed work has been significantly extended. Our new experimental results show that compared to MHGCN, our extended MHGCN+ significantly improves the performance on link prediction and node classification by 7.47% and 3.24% in terms of F1 score and Macro-F1 score across all datasets, respectively. The differences between this work and the conference paper are summarized as follows:

- We extend MHGCN by exploring the more effective multiplex relation aggregation methods for AMHENs and optimizing the multilayer graph convolution module.
- We conduct additional experiments to demonstrate the effectiveness of the extended MHGCN+ framework on one new real-world multiplex network. Results show MHGCN+ is significantly better than MHGCN in the previous conference. The new experimental results are shown in Table 3 and Table 4.
- We add the ablation study to verify the effectiveness of the proposed four independent aggregations. We also visualize the learned relation weights to illustrate the difference between MHGCN+ and MHGCN. Experimental results are shown in Figure 6, Figure 7 and Figure 8.
- We also re-perform the experiments for parameter sensitivity of the extended MHGCN+. Experimental results are shown in Figure 10.
- We also add and discuss the differences and connections between our model and the recent related works in Sec. 5.

2 RELATED WORK

Graph Neural Networks. The goal of a GNN is to learn a low-dimensional vector representation for each node, which can be used for many downstream network mining tasks. Kipf *et al.* [17] propose to perform convolutional operations over neighboring nodes for information aggregation. GraphSAGE [9] is an inductive GNN framework, which uses the general aggregating functions for the efficient generation of node embeddings. To differentiate the influence of neighboring nodes, GAT [33] is proposed as an attentive message-passing mechanism to learn the explicit weights of neighbor node embeddings. R-GCN [28] considers the influence of different edge types on nodes, and uses weight sharing and coefficient constraints to apply to multi-graphs with large numbers of relations. To simplify graph convolutional network, SGC [39] removes nonlinear transformation and training parameters in GCN, and LightGCN [11] omits the embedding projection with non-linearity during the message passing. Additionally, AM-GCN [36] is proposed to adaptively learn deep correlation information between topological structures and node features. However, all algorithms mentioned above are developed for homogeneous networks, and thus cannot effectively preserve the heterogeneous and multiplex graph characteristics for the network representation task.

Heterogeneous Graph Representation. Modeling the heterogeneous context of graphs has already received some attention [5, 23, 30, 49]. For example, some works leverage random walk sampling to construct meta-paths over the heterogeneous graph for node embeddings, including metapath2vec [5] and HERec [30]. As GNN has become a popular choice for encoding graph structures, many heterogeneous graph neural network models are designed to enhance the GNN architecture with the capability of capturing the node and edge heterogeneous contextual signals. For example, HetGNN [49] jointly encodes the graph topology and context heterogeneity with an attention mechanism for node embedding. HAN [34] extends GAT to heterogeneous networks through meta-path based neighbor discovery strategy and hierarchical attention mechanism. HeGAN [12] incorporates generative adversarial networks (GAN) for heterogeneous network representation learning. NARS [45] first generates relation subgraphs and learns node embeddings by 1D convolution on the generated subgraphs, and then aggregates the learned embeddings. Fu *et al.* [7] perform both the intra- and inter-metapath aggregation so as to distill the meta-path-based relational context for learning node representations. However, most of those approaches rely on selecting useful meta-paths to guide the process of heterogeneous representation learning, which may need external domain knowledge for constructing appropriate meta-paths.

In addition, there exist some recent studies [31, 37] attempting to relax the requirement of meta-path construction for heterogeneous network embedding. For example, HGT [13] proposes to incorporate self-attention into the graph-based message-passing mechanism for modeling the dynamic dependencies among heterogeneous nodes. HPN [15] eliminates semantic confusion by mapping nodes in meta-path to semantic space, and then aggregates the embeddings of nodes under different meta-paths to obtain the final representation. However, most of the above heterogeneous graph embedding models ignore the multiplex relational context of real-life graph data, in which multi-typed relationships exist among nodes.

Multiplex Heterogeneous Network Embedding. Real-world graphs are often inherently multiplex, which involves various relations and interactions between two connected nodes. To tackle this challenge, many multiplex network embedding techniques are proposed to project diverse node edges into latent representations. In particular, PMNE [19] uses three general aggregation models to obtain one overall node embedding for multiplex networks. MNE [50] introduces a global transformation matrix for each layer of the network to align the embeddings with different dimensions for each relation type. GATNE [2] learns base embedding, edge embedding as well as attribute embedding to generate the overall node representation. Motivated by the mutual information maximization scheme, DMGI [25] is proposed to minimize the difference among relation-aware node representations in an unsupervised learning manner. HGSL [53] first obtains the node representation based on meta-paths, and then uses GNN to obtain the final node embedding through heterogeneous graph structure learning optimization. However, the generality of the above methods is limited by their manual construction of meta-paths.

Recently, FAME [21] proposes a spectral graph transformation component to aggregate information from sub-networks by preserving relation-aware node dependencies. However, this model is built on random projection and sacrifices the adaptive parameter learning in exchange for fast embedding projection. GTN [48] computes the convex combinations of adjacency matrices of sub-networks with different weights to obtain candidate adjacency matrices to generate useful meta-paths. Furthermore, to learn the node embeddings of multiplex bipartite graph, DualHGCN [43] firstly generates two sets of homogeneous hypergraphs and then performs the information propagation with the spectral hypergraph convolutions. Jing *et al.* [16] propose the HDMI framework that explores the high-order mutual information to construct the supervision signals for learning node embedding on multiplex networks.

Although some previous methods can directly or indirectly use meta-paths to learn the characteristics of heterogeneity in heterogeneous networks, it can be seen that most of these methods need to specify special meta-paths artificially. When the selected meta-path is inappropriate, the performance of the methods will be greatly affected. To solve this problem, We want to design a model to learn the importance of various meta-paths adaptively, thereby avoiding human influence and making the method more stable.

3 PROBLEM DEFINITION

Table 1. Main notations and their definitions.

Notation	Definition
$\mathcal{G}$	the input network
$\mathcal{V}, \mathcal{E}$	the node/edge set of $\mathcal{G}$
$\mathcal{O}, \mathcal{R}$	the node/edge type set of $\mathcal{G}$
$\mathbf{X}$	the node attribute matrix of $\mathcal{G}$
$\mathcal{G}_r$	the sub-network *w.r.t.* edge type r
$\mathbf{A}_r$	the adjacency matrix of $\mathcal{G}_r$
$\mathbb{A}$	the aggregated adjacency matrix
$\hat{\mathbf{A}}_l$	the aggregated adjacency matrix of l-length meta-paths
$\mathbf{H}^{(l)}$	the hidden representation for the l-th layer
$\mathbf{H}$	the node embeddings
d	the dimension of embeddings
n, m	the number of nodes/attributes
$\mathbf{W}^{(l)}$	the learnable weight matrix for the l-th layer

In this section, we first introduce key concepts used in this paper and then formally define the studied problem.

We define graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with the set of nodes $\mathcal{V}$ and edges $\mathcal{E}$. Each edge in $\mathcal{E}$ represents a relationship between two nodes.

Definition 1 (Attributed Multiplex Heterogeneous Network, or AMHEN). *An attributed heterogeneous network is a network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$ associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{O}$ and an edge type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathbf{X} \in \mathbb{R}^{n \times m}$ is the attribute feature matrix, and $\mathcal{O}$ and $\mathcal{R}$ are the set of all node types and the set of all edge types, respectively. Each node $v \in \mathcal{V}$ belongs to a particular node type, and each edge $e \in \mathcal{E}$ is categorized into a specific edge type. With the consideration of node and edge heterogeneity, if $|\mathcal{O}| + |\mathcal{R}| > 2$, the network is **heterogeneous**. Additionally, with the consideration of edge multiplexity, if multiple types of edges exist between the same node pairs, the network is **attributed multiplex heterogeneous**.*

Definition 2 (Meta-path). *A meta-path $\mathcal{P}$ is defined as a path in the form of $O_1 \xrightarrow{r_1} O_2 \xrightarrow{r_2} \cdots \xrightarrow{r_{l-1}} O_l$ which describes a composite relation $R = r_1 \circ r_2 \cdots r_{l-1}$ between node types O_1 and O_l, where $\circ$ denotes the composition operator on relations.*

For example, $User \xrightarrow{click} Item \xrightarrow{buy} User$ is a 2-length meta-path, and $U_1 \xrightarrow{click} I_2 \xrightarrow{buy} U_2$ is a meta-path sample of the meta-path $User \xrightarrow{click} Item \xrightarrow{buy} User$.

Based on the above definitions, we formally present the representation learning task over the multiplex heterogeneous graph as follows:

Problem (Attributed Multiplex Heterogeneous Graph Representation). *The objective of our representation learning task over the attributed multiplex heterogeneous graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$ is to learn a low-dimensional latent embedding (with the hidden dimensionality of d, $d \ll |\mathcal{V}|$) for each node $v \in \mathcal{V}$, with the preservation of node and edge heterogeneity and multiplexity.*

The key notations used in this work are summarized in Table 1.

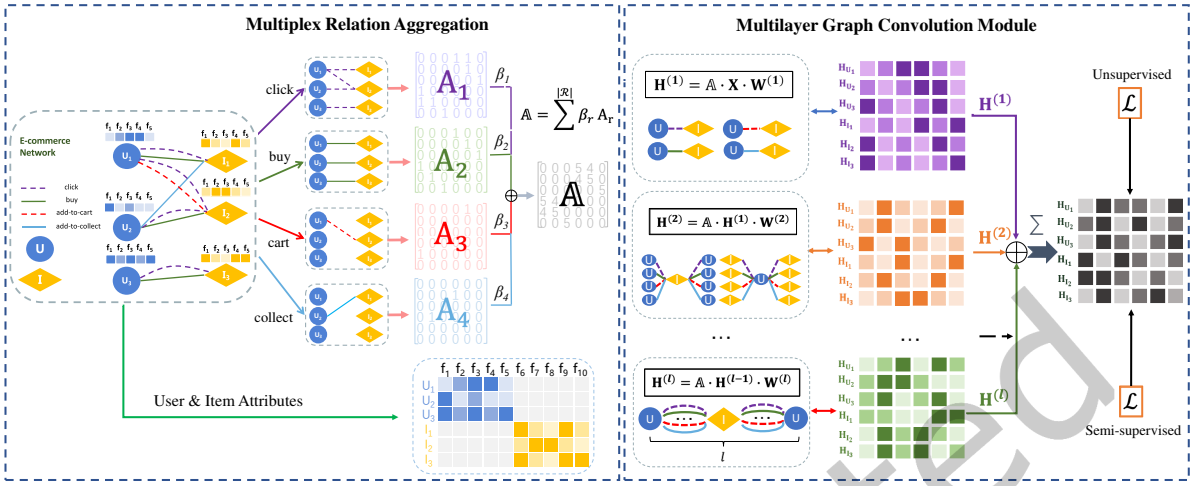


Fig. 1. The overview of the proposed MHGCN.

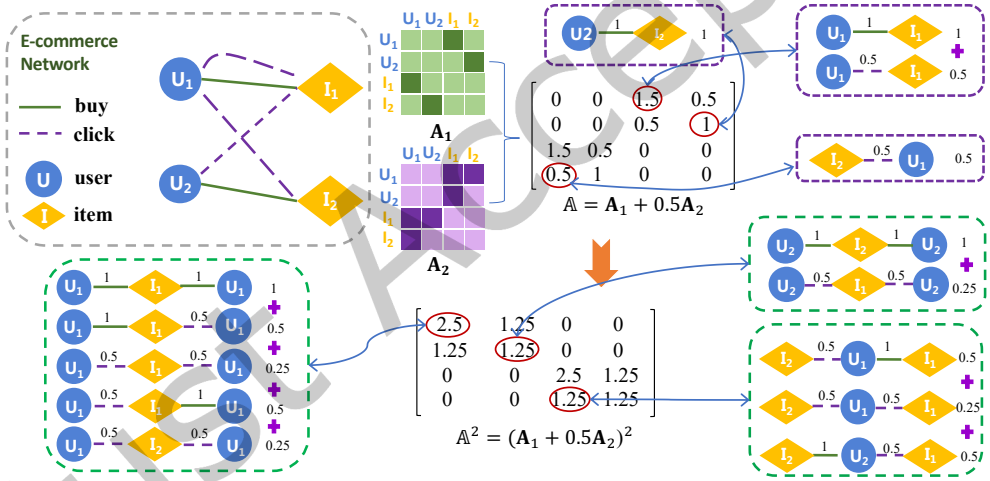


Fig. 2. Illustration of meta-paths with importance for a toy example

4 METHODOLOGY

In this section, we present our framework MHGCN with the overall architecture shown in Figure 1. Specifically, our MHGCN consists of two key components: (i) *multiplex relation aggregation* and (ii) *multilayer graph convolution module*. *Multiplex relation aggregation* aims to aggregate the multi-relations among heterogeneous nodes in multiplex heterogeneous networks by differentiating each relation with importance. *Multilayer graph convolution module* can automatically capture the heterogeneous meta-paths of different lengths across multi-relations by aggregating neighboring nodes' characteristics to learn the low-dimensional representation of nodes.

4.1 Multiplex Relation Aggregation

As defined in Sec. 3, there exist different types of nodes and multiple types of edges between these nodes in AMHENs, and each type of edge has a different role and impact on node representation. Therefore, following [21], we first generate multiple sub-graphs by differentiating the types of edge connections between nodes in the multiplex and heterogeneous graph. Thereafter, we aggregate the relation-aware graph contextual information with different importance weights.

We denote our generated sub-graphs as $\{\mathcal{G}_r|r = 1, 2, \ldots, |\mathcal{R}|\}$ with the corresponding adjacent matrix $\{\mathbf{A}_r|r = 1, 2, \ldots, |\mathcal{R}|\}$. Considering the scenario of multiplex user-item relations in online retailers (*e.g.*, click, purchase, add-to-cart), the decomposed sub-graph corresponds to the individual type of relationship between user and item. For instance, for graph representation learning in E-commerce platforms, different relationships (different edge types) between user and item nodes exhibit various dependency semantics. For example, the diverse behaviors of users (*e.g.*, click, add-to-favorite, purchase) reflect different preferences of users over items. Hence, multiplex user-item interactions with various relation semantics will have different impacts on the learning process of user representations. To capture such multi-typed node dependencies, our proposed MHGCN learns the relation-aware weights β_r to aggregate edge-type-specific sub-graph adjacent matrix as: $\mathbb{A} = \sum_{r=1}^{|\mathcal{R}|} \beta_r \mathbf{A}_r$. Notice that the set of weights $\{\beta_r|r = 1, 2, \ldots, |\mathcal{R}|\}$ should not be a set of hyperparameters but should be dynamically changed according to different tasks, so we set them as trainable parameters to be learned in model training.

4.2 Multilayer Graph Convolution Module

Different from homogeneous networks, heterogeneous networks contain different types of nodes and edges. The specified types of edges and nodes form a meta-path, which has an obvious effect on the representation learning of heterogeneous networks. Previous works require manually defined meta-paths and learning node representations on the sampled heterogeneous meta-paths. However, setting and sampling meta-paths artificially is a complex task. In a large-scale network, the number of meta-paths is very large. It takes a long time to sample such a large number of meta-paths. At the same time, aggregating meta-paths into meta-path graph also requires a lot of memory overhead. Additionally, the type of meta-paths has an important impact on node representation, which almost determines the performance of network embedding in various downstream tasks. The number of types of heterogeneous meta-paths is also very large, involving different lengths and different relation interactions. Therefore, it is difficult to select the appropriate meta-path types for heterogeneous network embedding methods based on meta-path aggregation. Our MHGCN effectively solves the above problems. We now present our multilayer graph convolution module that automatically captures the short and long meta-paths across multi-relations in AMHENs.

It is worth noting that our model employs a multi-layer fusion GCN. As shown in Figure 1, our graph convolution module consists of multiple graph convolutional layers. Its purpose is to capture meta-path information of different lengths. Next, we take a two-layer GCN as an example to illustrate how our model captures meta-path information. For a single-layer GCN:

$$\mathbf{H}^{(1)} = \mathbb{A} \cdot \mathbf{X} \cdot \mathbf{W}^{(1)}, \tag{1}$$

where $\mathbf{H}^{(1)} \in \mathbb{R}^{n \times d}$ is the output of first layer (*i.e.*, hidden representation of network), $\mathbf{X} \in \mathbb{R}^{n \times m}$ is the node attribute matrix, and $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times d}$ is the learnable parameter matrix. Notice that our convolution adopts the idea of SGC [39], that is, no non-linear activation function is used.

For the two-layer GCN, the message-passing process can be represented as below:

$$\begin{aligned} \mathbf{H}^{(2)} &= \mathbb{A} \cdot \mathbf{H}^{(1)} \cdot \mathbf{W}^{(2)} \\ &= \mathbb{A} \cdot (\mathbb{A} \cdot \mathbf{X} \cdot \mathbf{W}^{(1)}) \cdot \mathbf{W}^{(2)} \\ &= \mathbb{A}^2 \cdot \mathbf{X} \cdot \mathbf{W}^{(1)} \cdot \mathbf{W}^{(2)}, \end{aligned} \tag{2}$$

where $\mathbf{W}^{(2)} \in \mathbb{R}^{d \times d}$ is the learnable parameter matrix for the second layer.

Figure 2 illustrates a toy example of an E-commerce network, where we only consider two relations (*e.g.*, buy and click) between user and item nodes in this case. As shown in Figure 2, the aggregated matrix $\mathbb{A}$ can be regarded as a meta-path matrix generated by the 1-length meta-paths with importance (*i.e.*, all connected node pairs across all edge types with weights). For example, $\mathbb{A}_{(1,3)} = 1.5$ represents two 1-length meta-path samples with weights, *i.e.*, $U_1 \xrightarrow{1*buy} I_1 : 1$ and $U_1 \xrightarrow{0.5*click} I_1 : 0.5$. Therefore, the single-layer GCN can effectively learn the node representation that contains 1-length meta-path information. Similarly, the second power of $\mathbb{A}$ automatically captures the 2-length meta-path information with importance weights for all node pairs. For example, $\mathbb{A}^2_{(1,1)} = 2.5$ implies five 2-length meta-path samples across multi-relations with importance, *i.e.*, $U_1 \xrightarrow{1*buy} I_1 \xrightarrow{1*buy} U_1 : 1$, $U_1 \xrightarrow{1*buy} I_1 \xrightarrow{0.5*click} U_1 : 0.5$, $U_1 \xrightarrow{0.5*click} I_1 \xrightarrow{0.5*click} U_1 : 0.25$, $U_1 \xrightarrow{0.5*click} I_1 \xrightarrow{1*buy} U_1 : 0.5$, and $U_1 \xrightarrow{0.5*click} I_2 \xrightarrow{0.5*click} U_1 : 0.25$. The sum of the importance of these five meta-path samples is 2.5. Therefore, in Eq. (2), two-layer GCN can capture the 2-length meta-path information in $\mathbf{H}^{(2)}$.

At the same time, considering that the influence of meta-paths with different lengths on node embedding should also be different, the learnable parameter matrices $\mathbf{W}^{(l)}$ in our multilayer graph convolution module can just play this role. Eventually, we fuse the outputs of single-layer GCN and two-layer GCN:

$$\mathbf{H} = \frac{1}{2}(\mathbf{H}^{(1)} + \mathbf{H}^{(2)}). \tag{3}$$

The embedding $\mathbf{H} \in \mathbb{R}^{n \times d}$ contains all 1-length and 2-length meta-path information.

To capture longer heterogeneous meta-paths, we can extend it to l-layer:

$$\begin{aligned}
\mathbf{H}^{(l)} &= \mathbb{A} \cdot \mathbf{H}^{(l-1)} \cdot \mathbf{W}^{(l)} \\
&= \mathbb{A} \cdot (\mathbb{A} \cdot \mathbf{H}^{(l-2)} \cdot \mathbf{W}^{(l-1)}) \cdot \mathbf{W}^{(l)} \\
&= \underbrace{\mathbb{A} \cdots (\mathbb{A}}_{l} \cdot \mathbf{X} \cdot \underbrace{\mathbf{W}^{(1)}) \cdots \mathbf{W}^{(l)}}_{l} \\
&= \mathbb{A}^l \cdot \mathbf{X} \cdot \underbrace{\mathbf{W}^{(1)} \cdots \mathbf{W}^{(l)}}_{l}
\end{aligned} \tag{4}$$

Finally, we fuse outputs of all layers to capture all meta-path information of different lengths across multi-relations:

$$\mathbf{H} = \frac{1}{l} \sum_{i=1}^{l} \mathbf{H}^{(i)}. \tag{5}$$

4.3 Model Learning

This section presents the objective function to train our model to learn the final node representation. Depending on the requirements of different downstream tasks and the availability of node labels, we can train MHGCN in two major learning paradigms, *i.e.*, unsupervised learning and semi-supervised learning.

For unsupervised learning, we minimize the following binary cross-entropy loss function through negative sampling to optimize the model parameters:

$$\begin{aligned}\mathcal{L}_{usl} = &- \sum_{(u,v)\in\Omega} \log\sigma(< \mathbf{H}_u^{\mathsf{T}}, \mathbf{H}_v >) \\ &- \sum_{(u',v')\in\Omega^-} \log\sigma(- < \mathbf{H}_{u'}^{\mathsf{T}}, \mathbf{H}_{v'} >),\end{aligned} \tag{6}$$

where $\mathbf{H}_v$ is the representation of node v, T denotes matrix transposition, $\sigma(\cdot)$ is the sigmoid function, $<,>$ is the vector similarity measure function (*e.g.*, inner product), Ω is the set of positive node pairs, Ω^- is the set of negative node pairs sampled from all unlinked node pairs.

For semi-supervised node classification, we can optimize the model parameters by minimizing the cross entropy via backpropagation and gradient descent. The cross-entropy loss over all labeled nodes between the ground truth and the prediction is formulated as:

$$\mathcal{L}_{ssl} = - \sum_{i\in\mathcal{V}_{ids}} \mathbf{Y}_i \ln(\mathbf{C}\cdot\mathbf{H}_i), \tag{7}$$

where $\mathcal{V}_{ids}$ is the set of node indices that have class labels, $\mathbf{Y}_i$ is the label of the i-th node, $\mathbf{C}$ is the node classifier parameter, and $\mathbf{H}_i$ is the representation of the i-th node. With the guide of a small fraction of labeled nodes, we can optimize the proposed model and then learn the embeddings of nodes for semi-supervised classification.

Notice that $\{\mathbf{W}^{(i)}|i = 1, 2, \ldots, l\}$ and $\{\beta_r|r = 1, 2, \ldots, |\mathcal{R}|\}$ in our model can be learned during training phase. The pseudo-code of our proposed MHGCN is shown in Algorithm 1.

Algorithm 1 The Learning Process of MHGCN

Input: Input AMHEN $\mathcal{G}$, node feature matrix $\mathbf{X}$, embedding dimension d, the number of convolution layers l
Output: Embedding results $\mathbf{H}$

```
Decouple the attributed multiplex heterogeneous network into homogeneous networks and bipartite networks to obtain the adjacency matrices {A_r|r = 1, 2, ..., |R|}
Calculate 𝔸 = Σ_{r=1}^{|R|} β_r A_r
for i = 1 to l do
    Calculate H^(i) ← 𝔸 · H^(i−1) · W^(i)
end for
H = (1/l)(H^(1) + ··· + H^(l))
Calculate ℒ using Eq. (6) or Eq. (7);
Back propagation and update parameters in MHGCN
Return H
```

5 CONNECTION WITH PREVIOUS WORKS

The works most relevant to our proposed model are FAME [21] and GTN [48], and thus we discuss the differences and connections between our model and them in this section.

5.1 FAME

Multiplex relation aggregation of our MHGCN is inspired by FAME. FAME mainly includes two key components: spectral graph transformation and fast random projection embedding.

The spectral graph transformation process is similar to our multiplex relation aggregation $\mathbb{A} = \sum_{r=1}^{|\mathcal{R}|} \beta_r \mathbf{A}_r$, but the weights β_r of different relations in FAME are given in the form of hyperparameter. To capture the high-order proximities between nodes, a spectral graph transformation on $\mathbb{A}$ is performed: $F(\mathbb{A}) = \sum_{i=1}^{K} \alpha_i \mathbb{A}^i$, where α_i is the weight for the i-th order proximity, and K is the highest order. Both weights β_r and α_i are set as hyperparameters in FAME, which are tuned by optuna [1], a Bayesian hyperparameter optimization method.

In the process of random projection embedding, FAME first obtains a random projection matrix $\mathbf{R}$, and then incorporates $F(\mathbb{A})$ with node attribute features and finally gets the low-dimensional embeddings of nodes by random projection:

$$\begin{aligned} \mathbf{H} &= F(\mathbb{A}) \cdot \mathbf{X} \cdot \mathbf{R} \\ &= \sum_{i=1}^{K} \alpha_i \mathbb{A}^i \cdot \mathbf{X} \cdot \mathbf{R} \\ &= \sum_{i=1}^{K} \alpha_i (\sum_{r=1}^{|\mathcal{R}|} \beta_r \mathbf{A}_r)^i \cdot \mathbf{X} \cdot \mathbf{R}. \end{aligned} \tag{8}$$

Since the weights β_r and α_i in FAME are given in the form of hyperparameters, its multiplex aggregation effect may not be optimal or sub-optimal. Therefore, in our MHGCN, we set the aggregation weights β_r as learnable parameters to obtain the importance of different relations through model training, so as to achieve a better aggregation effect.

In addition, it can be seen that the aggregated adjacency matrices composed of meta-paths of different lengths are not projected respectively, but share the same random projection matrix $\mathbf{R}$. To achieve better dimensionality reduction, we use a learnable weight matrix to replace the random projection matrix, so as to obtain better node embedding. Moreover, we also find that the most suitable weight matrices for node embeddings obtained by meta-paths of different lengths are not the same. Therefore, in our MHGCN, for each length of meta-paths, *i.e.*, each order i, we set a different learnable weight matrix $\mathbf{W}^{(i)}$. It should be noted that since we set a different weight matrix $\mathbf{W}^{(i)}$ for each layer, it is not necessary to set the parameters α_i like FAME, because α_i can be absorbed by $\mathbf{W}^{(i)}$ to achieve the same effect.

5.2 GTN

GTN reconstructs a multiplex heterogeneous information network into a new graph composed of meta-paths of various lengths and then uses GCN to learn node embedding from the new graph. GTN can also be divided into two stages: meta-path generation and graph transformer networks.

In the meta-path generation stage, similar to our multiplex relation aggregation, GTN first decouples multiplex heterogeneous networks into multiple sub-networks, each of which contains only one relation. Then, meta-paths of different lengths can be obtained by the multiplications of the aggregated adjacency matrices:

$$\mathbf{A}^{(l)} = (\sum_{r=0}^{|\mathcal{R}|} \beta_{1,r} \mathbf{A}_r) \cdots (\sum_{r=0}^{|\mathcal{R}|} \beta_{l,r} \mathbf{A}_r) \tag{9}$$

where $\mathbf{A}_0$ is an identity matrix, and $\beta_{i,r}$ denotes the weight for r-th relation at i-th step of meta-path. Therefore, $\mathbf{A}^{(l)}$ represents the graph obtained by aggregating all meta-paths from length 0 to length l with different weights.

It can be seen that each aggregation in GTN uses an independent set of weights, while the same set of weights β_r is used in MHGCN. We will discuss this difference in the next section.

After obtaining the adjacency matrix of meta-paths, *i.e.*, $\mathbf{A}^{(l)}$, GTN feeds $\mathbf{A}^{(l)}$ into GCN to obtain node embeddings:

$$\mathbf{H} = \sigma(\tilde{\mathbf{D}}^{(-1)} \cdot \tilde{\mathbf{A}}^{(l)} \cdot \mathbf{X} \cdot \mathbf{W}), \tag{10}$$

where $\tilde{\mathbf{A}}^{(l)} = \mathbf{A}^{(l)} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}^{(l)}$.

It can be seen that in GTN, similar to FAME, the same weight matrix (*i.e.*, $\mathbf{W}$) is used for meta-paths of different lengths, which limits the improvement of its performance. To address the defects of GTN, MHGCN firstly considers the influence and importance of meta-paths of different lengths on node representation learning, instead of the same GCN network. Additionally, to further improve efficiency, MHGCN also adopts the idea of simplifying GCN, removing the non-linear activation function.

6 MODEL EXTENSION

In this section, we first analyze the multiplex relation aggregation module and then extend the module to improve model performance. According to the independence between different meta-paths, we divide the multiplex relation aggregation into four levels: *primary independent aggregation*, *secondary independent aggregation*, *tertiary independent aggregation*, and *fully independent aggregation*.

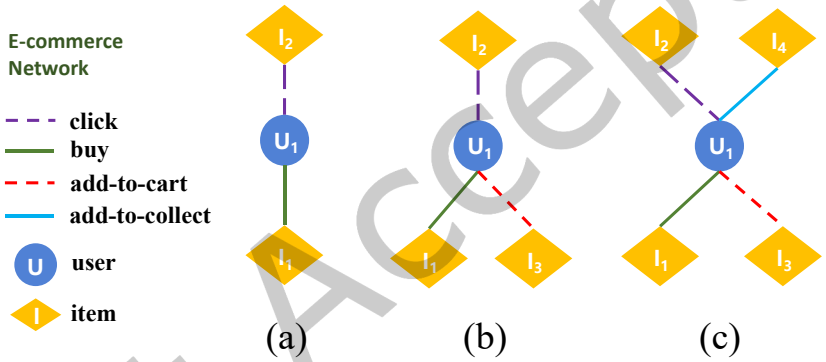


Fig. 3. Illustration of meta-paths with conflicting importance

6.1 Primary Independent Aggregation

Primary independent aggregation refers to the fact that the same set of weights is used for different relations in meta-paths during the multiplex relation aggregation:

$$\hat{\mathbf{A}}_l = \mathbb{A}^l = (\sum_{r=1}^{|\mathcal{R}|} \beta_r \mathbf{A}_r)^l. \tag{11}$$

where $\hat{\mathbf{A}}_l$ represents the aggregated adjacency matrix of l-length meta-paths. This aggregation is adopted in both FAME and our MHGCN. In this aggregation, $|\mathcal{R}|$ learnable parameters are used to aggregate different types of relations to obtain the combined adjacency matrix $\mathbb{A}$, which represents the adjacency matrix by aggregating all 1-length meta-paths with different importance.

The biggest advantage of this aggregation is that it introduces fewer learnable parameters. Additionally, the impact of overfitting during training is also minimal.

Nonetheless, this weight setting may bring conflicting issues regarding the importance of different meta-paths. For example, as shown in Figure 3(a), if both purchase and click relations are important in the 1-length meta-paths, *i.e.*, the value of β_{buy} and β_{click} are going to be large. Since the weight of the longer meta-path is derived by multiplying the weight of the 1-length meta-path, which makes the importance of the 2-length meta-path $I \xrightarrow{buy} U \xrightarrow{click} I$, $\beta_{buy} * \beta_{click}$, must also be large.

6.2 Secondary Independent Aggregation

Secondary independent aggregation refers to the aggregation of multiplex relations with independent weight settings at different steps of meta-paths, which is defined as:

$$\begin{cases} \hat{\mathbf{A}}_1 = (\sum_{r=1}^{|\mathcal{R}|} \beta_{1,r} \mathbf{A}_r) \\ \hat{\mathbf{A}}_2 = (\sum_{r=1}^{|\mathcal{R}|} \beta_{2,r} \mathbf{A}_r) \hat{\mathbf{A}}_1 \\ \quad \vdots \\ \hat{\mathbf{A}}_l = (\sum_{r=1}^{|\mathcal{R}|} \beta_{l,r} \mathbf{A}_r) \hat{\mathbf{A}}_{l-1} \end{cases} \tag{12}$$

In the secondary independent aggregation, there are $l * |\mathcal{R}|$ learnable parameters for capturing meta-paths of maximum l length.

Secondary independent aggregation is an extension of the primary independent aggregation, which can effectively address the conflicting problem in the primary independent aggregation. For instance, even though both purchase and click relations are important in 1-length meta-paths (*i.e.*, both $\beta_{1,buy}$ and $\beta_{1,click}$ are large), the 2-length meta-path $I \xrightarrow{buy} U \xrightarrow{click} I$ can also get a lower weight by setting independent lower weight $\beta_{2,click}$ for click relation on the second step.

However, secondary independent aggregation still has certain limitations. As shown in Figure 3(b), when purchase relation is important (*i.e.*, $\beta_{1,buy}$ is large), but add-to-cart relation is not (*i.e.*, $\beta_{1,cart}$ is small) in the 1-length meta-paths, and the 2-length meta-path $I \xrightarrow{cart} U \xrightarrow{click} I$ is important, the weight $\beta_{2,click}$ must be large, which leads to the greater importance of the 2-length meta-path $I \xrightarrow{buy} U \xrightarrow{click} I$ (*i.e.*, $\beta_{1,buy} * \beta_{2,click}$). Namely, secondary independent aggregation can solve the weight conflict between 1-length meta-paths and 2-length meta-paths, but it cannot solve the weight conflict between two 2-length meta-paths.

GTN adopts the secondary independent aggregation and effectively extends secondary independent aggregation by introducing an identity matrix at each multiplex relation aggregation on each step.

6.3 Tertiary Independent Aggregation

The tertiary independent aggregation is extended from the secondary independent aggregation, so that the weights used for meta-paths of different lengths are completely independent, which can effectively solve the

weight-conflicting problem in the secondary independent aggregation:

$$\begin{cases} \hat{\mathbf{A}}_1 = (\sum_{r=1}^{|\mathcal{R}|} \beta_{1,1,r} \mathbf{A}_r) \\ \hat{\mathbf{A}}_2 = \prod_{i=1}^{2} (\sum_{r=1}^{|\mathcal{R}|} \beta_{2,i,r} \mathbf{A}_r) \\ \quad \vdots \\ \hat{\mathbf{A}}_l = \prod_{i=1}^{l} (\sum_{r=1}^{|\mathcal{R}|} \beta_{l,i,r} \mathbf{A}_r) \end{cases} \tag{13}$$

As depicted in Eq. (13), there are $\sum_{i=1}^{l} i * |\mathcal{R}|$ learnable parameters for aggregating different relations to obtain the combined adjacency matrices. Specifically, the sets of weights $\{\beta_{l,i,r} | i = 1, \ldots, l, r = 1, \ldots, |\mathcal{R}|\}$ used in different adjacency matrices $\hat{\mathbf{A}}_l$ are independent of each other, rather than the weights of shorter meta-paths inherited by longer meta-paths like secondary independent aggregation.

In the tertiary independent aggregation, since the weights of the longer meta-paths are not limited by the weights of the shorter meta-paths, the weight conflict in Figure 3(b) can be solved.

Although tertiary independent aggregation solves the weight-conflicting problem in secondary independent aggregation, it still cannot fundamentally solve the conflicting problem of different meta-paths. For example, as shown in Figure 3(c), there are four 2-length meta-paths, *i.e.*, $I \xrightarrow{buy} U \xrightarrow{click} I$, $I \xrightarrow{buy} U \xrightarrow{collect}$, $I \xrightarrow{cart} U \xrightarrow{click} I$, and $I \xrightarrow{cart} U \xrightarrow{collect} I$. If $I \xrightarrow{buy} U \xrightarrow{click} I$ and $I \xrightarrow{cart} U \xrightarrow{collect} I$ are very important, while $I \xrightarrow{buy} U \xrightarrow{collect} I$ and $I \xrightarrow{cart} U \xrightarrow{click} I$ are not in 2-length meta-paths. At this time, $I \xrightarrow{buy} U \xrightarrow{click} I$ (important) and $I \xrightarrow{cart} U \xrightarrow{click} I$ (not important) share $U \xrightarrow{click} I$, thus this requires that the weight $\beta_{2,1,buy}$ is relatively large, and the weight $\beta_{2,1,cart}$ is relatively small. Similarly, since $I \xrightarrow{buy} U \xrightarrow{collect} I$ (not important) and $I \xrightarrow{cart} U \xrightarrow{collect} I$ (important) shares $U \xrightarrow{collect} I$, this requires the weight $\beta_{2,1,buy}$ is relatively small, and the weight $\beta_{2,1,cart}$ is relatively large. It can be seen that the problem of weight conflict occurs again in this case.

In fact, this weight conflict is caused by our attempt to multiply two $1 * k$ vectors to simulate the $k * k$ matrix. Although the weight matrices of all 2-length meta-paths can be obtained by multiplying the two vectors, the weights of any two rows in the weight matrix have a strong linear relationship, which leads to the weight-conflicting problem of tertiary independent aggregation.

6.4 Fully Independent Aggregation

Fully independent aggregation is an aggregation method used to completely solve the weight conflict between different meta-paths. In the fully independent aggregation, various types of relations are not aggregated according to the weight first, but the adjacency matrices formed by various types of meta-paths are calculated first, and

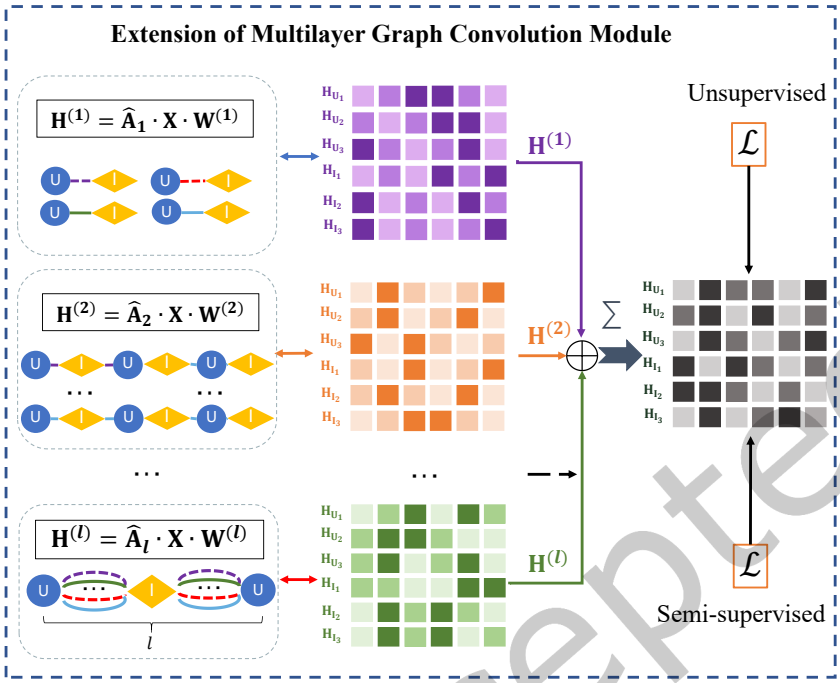


Fig. 4. The extended multilayer graph convolution module

then are aggregated according to their weights:

$$\begin{cases} \hat{\mathbf{A}}_1 = \sum_{i=1}^{|\mathcal{R}|} \beta_i \mathbf{A}_i \\ \hat{\mathbf{A}}_2 = \sum_{i=1}^{|\mathcal{R}|} \sum_{j=1}^{|\mathcal{R}|} \beta_{i,j} \mathbf{A}_i \mathbf{A}_j \\ \quad \vdots \\ \hat{\mathbf{A}}_l = \sum_{i=1}^{|\mathcal{R}|} \cdots \sum_{j=1}^{|\mathcal{R}|} \beta_{i,\cdots,j} \underbrace{\mathbf{A}_i \cdots \mathbf{A}_j}_{l} \end{cases} \tag{14}$$

Through fully independent aggregation, the weight dependency between different meta-paths can be eliminated, and thus the weight conflict between meta-paths can be completely solved.

In the fully independent aggregation, total $\sum_{i=1}^{l} |\mathcal{R}|^i$ learnable parameters are used to aggregate different relations to obtain the combined adjacency matrices, and the weights of all meta-paths of different lengths are independent of each other, so as to solve the three weight conflicts mentioned above.

6.5 Extension of Multilayer Graph Convolution

Figure 4 shows the extended multilayer graph convolution module in MHGCN+. MHGCN adopts the primary independent aggregation. To capture the heterogeneous meta-path information of different lengths, the multilayer graph convolution module contains multiple GCNs of different layers. In MHGCN+, we employ the fully independent aggregation, which has obtained the adjacency matrices of heterogeneous meta-paths of different lengths. Therefore, for MHGCN+, the multilayer graph convolution module is extended to consist of multiple one-layer GCNs, *i.e.*, we can obtain node representations that fuse heterogeneous meta-paths of different lengths through multiply one-layer GCNs:

$$\begin{aligned} \mathbf{H}^{(l)} &= \hat{\mathbf{A}}_l \cdot \mathbf{X} \cdot \mathbf{W}^{(l)}, \\ \mathbf{H} &= \frac{1}{l} \sum_{i=1}^{l} \mathbf{H}^{(i)}, \end{aligned} \tag{15}$$

where $\mathbf{H}^{(l)}$ is the hidden representation corresponding to $\hat{\mathbf{A}}_l$, *i.e.*, the aggregated adjacency matrix of l-length meta-paths.

Table 2. Statistics of Datasets

Dataset	#nodes	#edges	#node types	#edge types	#features	Multiplex network
Retailrocket	4012	20000	2	3	4012	✓
Alibaba	21,318	41,676	2	4	19	✓
Amazon	10,166	148,865	1	2	1,156	✓
AMiner	58,068	118,939	3	3	8	×
IMDB	12,772	18,644	3	2	1,256	×
DBLP	26,128	119,783	4	3	4,635	×
AMiner-M	10,000	2,412,266	1	3	8	✓

7 EXPERIMENT

7.1 Datasets

Seven publicly available real-world datasets are used in experimental evaluation. Alibaba dataset[1] includes four types of edges between user and item nodes. We use the category of items as the class label in node classification. Amazon dataset[2] includes one node type of products in the Electronics category, and co-viewing and co-purchasing links between products. AMiner dataset[3] is a citation network, which contains three types of nodes: author, paper, and conference. The domain of papers is considered the class label. IMDB dataset[4] contains three types of nodes, *i.e.*, movie, actor, and director, and labels are genres of movies. Node features are given as bag-of-words representations of plots. DBLP dataset[5] contains four types of nodes, *i.e.*, author, paper, term, and venue. We use the authors' research field as a label for classification. Retailrocket dataset[6] is generated from

[1]https://tianchi.aliyun.com/competition/entrance/231719/information/
[2]http://jmcauley.ucsd.edu/data/amazon/
[3]https://github.com/librahu/
[4]https://github.com/seongjunyun/Graph_Transformer_Networks
[5]https://www.dropbox.com/s/yh4grpeks87ugr2/DBLP_processed.zip?dl=0
[6]https://www.kaggle.com/datasets/retailrocket/ecommerce-dataset

an online shopping site-Retailrocket over 4 months, to record three types of user behaviors, *i.e.*, purchase, page view, and add-to-cart. Based on AMiner dataset, we also construct a multiplex dataset, AMiner-M, including a type of paper nodes and three types of edges: co-author, co-conference, and reference. The domain of papers is considered as the class label. Since some baselines cannot scale to the whole Alibaba network, we evaluate all models on a sampled dataset from Alibaba. The statistics of these seven datasets are summarized in Table 2.

7.2 Baselines

We compare our model against the following eighteen graph embedding baselines, which are divided into three categories.

Homogeneous network embedding methods:

- **node2vec** [8] - node2vec is a network embedding method that samples short biased random walks.
- **RandNE** [52] - RandNE is a network embedding approach based on Gaussian random projection, which preserves the high-order proximities between nodes.
- **FastRP** [3] - FastRP is an extension of RandNE by using sparse random projection.
- **SGC** [39] - SGC is a simplified version of GCN, which only uses the product of high-order adjacency matrices and attribute matrix, without nonlinear transformation.
- **AM-GCN** [36] - AM-GNN is an adaptive multi-channel graph convolutional network for semi-supervised classification.

Heterogeneous network embedding methods:

- **R-GCN** [28] - R-GCN further considers the influence of different edge types on nodes, and uses weight sharing and coefficient constraints to apply to heterogeneous networks.
- **HAN** [34] - HAN applies graph attention network on multiplex network considering the inter- and intra-network interactions, which exploit manually selected meta-paths to learn node embedding.
- **NARS** [45] NARS decouples heterogeneous networks according to the type of edge, and then aggregates neighbor features on the decoupled subgraphs.
- **MAGNN** [7] - MAGNN is a meta-path aggregated graph neural network for heterogeneous graphs.
- **HPN** [15] - HPN designs a semantic propagation mechanism to alleviate semantic confusion and a semantic fusion mechanism to integrate rich semantics.

Multiplex Heterogeneous network embedding methods:

- **PMNE** [19] - PMNE contains three different models to merge the multiplex network to generate one overall embedding for each node, which are denoted as PMNE-n, PMNE-r, and PMNE-c, respectively.
- **MNE** [50] - MNE obtains the final embedding by combining the high-dimensional common embedding and the low-dimensional hierarchical embedding.
- **GATNE** [2] - GATNE includes two variants GATNE-T and GATNE-I. We use GATNE-I as our baseline method in experiments.
- **GTN** [48] - GTN transforms a heterogeneous graph into multiple meta-path graphs and then learns node embeddings via GCN on the meta-path graphs.
- **DMGI** [25] - DMGI integrates node embeddings from multiple graphs by introducing a consensus regularization framework and a universal discriminator.
- **FAME** [21] - FAME is a random projection-based network embedding for AMHENs, which uses spectral graph transformation to capture meta-paths, and improves efficiency through random projection.
- **HGSL** [53] - HGSL is a state-of-the-art heterogeneous GNN, which jointly performs heterogeneous graph structure learning and GNN parameter learning for classification.
- **DualHGNN** [43] - DualHGCN uses a dual hypergraph convolutional network to learn node embeddings for multiplex bipartite networks.

Table 3. Link prediction performance comparison of different methods on six datasets

Method	AMiner			Alibaba			IMDB			Amazon			DBLP			Retailrocket		
	R-AUC	P-AUC	F1	R-AUC	P-AUC	F1	R-AUC	P-AUC	F1	R-AUC	P-AUC	F1	R-AUC	P-AUC	F1	R-AUC	P-AUC	F1
node2vec	0.594	0.663	0.602	0.614	0.580	0.593	0.479	0.568	0.474	0.946	0.944	0.880	0.449	0.452	0.478	0.501	0.559	0.626
RandNE	0.607	0.630	0.608	0.877	0.888	0.826	0.901	0.933	0.839	0.950	0.941	0.903	0.492	0.491	0.493	0.504	0.513	0.498
FastRP	0.620	0.634	0.600	0.927	0.900	0.926	0.869	0.893	0.811	0.954	0.945	0.893	0.515	0.528	0.506	0.492	0.505	0.493
SGC	0.589	0.585	0.567	0.686	0.708	0.623	0.826	0.889	0.769	0.791	0.802	0.760	0.601	0.606	0.587	0.796	0.742	0.756
R-GCN	0.599	0.601	0.610	0.674	0.710	0.629	0.826	0.878	0.790	0.811	0.820	0.783	0.589	0.592	0.566	0.775	0.748	0.695
MAGNN	0.663	0.681	0.666	0.961	0.963	0.948	0.912	0.923	0.887	0.958	0.949	0.915	0.690	0.699	0.684	0.847	0.847	0.772
HPN	0.658	0.664	0.660	0.958	0.961	0.950	0.900	0.903	0.892	0.949	0.949	0.904	0.692	0.710	0.687	0.796	0.805	0.733
PMNE-n	0.651	0.669	0.677	0.966	0.973	0.891	0.674	0.683	0.646	0.956	0.945	0.893	0.672	0.679	0.663	0.605	0.544	0.689
PMNE-r	0.615	0.653	0.662	0.859	0.915	0.824	0.646	0.646	0.613	0.884	0.890	0.796	0.637	0.640	0.629	0.531	0.503	0.613
PMNE-c	0.613	0.635	0.657	0.597	0.591	0.664	0.651	0.634	0.630	0.934	0.934	0.868	0.622	0.625	0.609	0.567	0.521	0.672
MNE	0.660	0.672	0.681	0.944	0.946	0.901	0.688	0.701	0.681	0.941	0.943	0.912	0.657	0.660	0.635	0.635	0.574	0.632
GATNE	OOT	OOT	OOT	0.981	0.986	0.952	0.872	0.878	0.791	0.963	0.948	0.914	OOT	OOT	OOT	0.502	0.627	0.523
DMGI	OOM	OOM	OOM	0.857	0.781	0.784	0.926	0.935	0.873	0.905	0.878	0.847	0.610	0.615	0.601	0.555	0.502	0.651
FAME	0.687	0.747	0.726	0.993	0.996	0.979	0.944	0.959	0.897	0.959	0.950	0.900	0.642	0.650	0.633	0.646	0.634	0.636
DualHGNN	/	/	/	0.974	0.977	0.966	/	/	/	/	/	/	/	/	/	0.790	0.869	0.815
HGTN	0.695	0.732	0.721	0.989	0.992	0.983	0.951	0.947	0.895	0.960	0.950	0.897	0.687	0.702	0.675	0.836	0.854	0.778
MHGCN	0.711	0.753	0.730	0.997	0.997	0.992	0.967	**0.966**	0.959	0.972	**0.974**	0.961	0.718	0.722	0.703	0.839	0.908	0.839
MHGCN+	**0.811**	**0.853**	**0.786**	**0.999**	**0.999**	**0.999**	**0.970**	0.952	**0.964**	**0.973**	0.965	**0.965**	**0.899**	**0.954**	**0.899**	**0.903**	**0.965**	**0.903**

OOT: Out Of Time (36 hours). OOM: Out Of Memory. R-AUC: ROC-AUC, P-AUC: PR-AUC.

- **HGTN** [18] - HGTN uses hypergraphs to capture heterogeneous information in heterogeneous information networks.

For homogeneous network embedding methods and heterogeneous network embedding methods to deal with multiplex networks, we feed separate graphs with a single-layer view into them to obtain different node embeddings, then perform mean pooling to generate final node embedding. Since DualHGNN is designed only for multiplex bipartite networks, it can only work on Alibaba and Retailrocket networks.

7.3 Experimental Setting

For link prediction, we randomly sample 85%, 5%, and 10% of edges as training set, validation set, and test set. At the same time, we also randomly sample the same number of negative node pairs (*i.e.*, unlinked node pairs) to add into the training set, validation set, and test set. Notice that we predict each type of edge using all types of edges in datasets, and finally take the average of all edges as the final result. For node classification, we randomly take 80% of the nodes as the training set, 10% as the validation set, and 10% as the test set. Notice that we repeat each experiment 10 times to report average results.

Following [21], we set d to 200 for all the methods. For baselines, we use the source code released by their authors or OpenHGNN[7], and adopt the parameter settings recommended in their papers and fine-tune them to be optimal. For our MHGCN and MHGCN+, we set the number of convolution layers l to 2 (*i.e.*, the length of meta-paths l for MHGCN+), tune learning rate in $\{0.01, 0.05, 0.001, 0.005, 0.0001, 0.0005\}$, and weight-decay to 0.0005.

We evaluate the efficiency evaluation for all methods on a machine with Intel Xeon E5-2660 (2.2GHz) CPU, 80GB memory, and 2 × GeForce RTX 2080 (8G).

[7]https://github.com/BUPT-GAMMA/OpenHGNN

7.4 Link Prediction

We first evaluate the model performance on the unsupervised link prediction task by comparing our MHGCN and MHGCN+ with fifteen baselines. The experimental results are shown in Table 3, where the best is shown in bold and the best among baselines is underlined. The first seven baselines are homogeneous or heterogeneous network embedding methods, and the last nine are multiplex network embedding methods.

As we can see, MHGCN significantly outperforms all baselines in terms of all evaluation metrics on six datasets. To be specific, MHGCN achieves average gains of 5.68% F1 score in comparison to the best performed GNN baselines across all datasets (*i.e.*, FAME, MAGNN, and HPN). Our MHGCN realizes a high accuracy of more than 96% on three datasets (Alibaba, Amazon, and IMDB), especially more than 99% prediction performance on Alibaba network. This is because MHGCN automatically captures effective multi-relational topological structures through multiplex relation aggregation and multilayer graph convolution on the generated meta-paths across multiplex relations. Particularly, compared with GATNE and MAGNN, our model has achieved better results, showing the ability of our model to automatically capture meta-paths compared with manually setting meta-paths. FAME which uses spectral graph transformation to achieve the second-best performance on most datasets also verifies the ability of multiplex relation aggregation to automatically capture useful heterogeneous meta-paths. However, MHGCN obtains better performance than FAME on all networks as our MHGCN learns meaning node representations for AMHENs using multilayer graph convolution in a learning manner. Moreover, MHGCN also shows significant performance advantages on general heterogeneous networks (*e.g.*, IMDB and DBLP). This may be because MHGCN uses a weighted approach to differentiate the effects of different types of relations on node representation, which cannot be achieved by traditional meta-path sampling.

Additionally, our extended MHGCN+ also significantly improves MHGCN for link prediction on all tested datasets. In particular, except for the three datasets where MHGCN's prediction performance exceeds 96%, MHGCN+ improves MHGCN by 7.67%, 27.88%, and 7.63% in terms of F1 score on AMiner, DBLP, and Retailrocket, respectively. These results demonstrate that the proposed full independent aggregation used in MHGCN+ more effectively captures the importance of heterogeneous meta-paths of different lengths for node representation learning in both multiplex and heterogeneous networks.

7.5 Node Classification

We next evaluate the effectiveness of our model on the semi-supervised node classification task compared with state-of-the-art baselines. The results are reported in Table 4, where the best is shown in bold and the best among baselines is underlined. The first eight baselines are unsupervised embedding methods, and the rest are semi-supervised embedding methods.

We can see that MHGCN also achieves state-of-the-art performance on all tested heterogeneous networks. In particular, our MHGCN outperforms the state-of-the-art GNN model HGSL on average by 11.22% and 14.49% in terms of Macro-F1 and Micro-F1 across all datasets. This performance improvement achieved by our MHGCN is remarkable considering that reported performance gains for node classification in some recent work [7, 53] are usually around 2-4%. This experiment validates the benefits of our framework, which models the multi-relational structure and node properties of AMHENs using multiplex relation aggregation and labeled data-guided multilayer graph convolution module. Furthermore, we also observe that MHGCN performs much better than competitor methods on general heterogeneous networks with multi-typed nodes (*e.g.*, IMDB and DBLP), achieving 23.23% and 22.19% improvement in Macro-F1 and Micro-F1 on IMDB network. The possible reason behind this is that our MHGCN effectively learns node representations for classification by exploring all meta-path interactions across multiple relations with different importance (*i.e.*, weights), which is ignored by the heterogeneous network embedding approaches based on manually setting meta-path sampling as they cannot realize the importance of different relations within each selected meta-path.

Table 4. Node classification performance comparison of different methods on four datasets

Method	AMiner-M		Alibaba		IMDB		DBLP	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
node2vec	0.522 (0.0032)	0.532 (0.0051)	0.238 (0.0125)	0.347 (0.0093)	0.363 (0.0237)	0.382 (0.0703)	0.352 (0.0103)	0.351 (0.0112)
RandNE	0.641 (0.0074)	0.672 (0.0064)	0.319 (0.0170)	0.358 (0.0093)	0.373 (0.0143)	0.392 (0.0185)	0.351 (0.0153)	0.372 (0.0150)
FastRP	0.650 (0.0086)	0.690 (0.0074)	0.301 (0.0180)	0.392 (0.0119)	0.363 (0.0236)	0.381 (0.0140)	0.343 (0.0201)	0.375 (0.0199)
MNE	0.643 (0.0069)	0.686 (0.0045)	0.289 (0.0155)	0.390 (0.0021)	0.374 (0.0153)	0.382 (0.0680)	0.366 (0.0117)	0.384 (0.0109)
GATNE	0.772(0.0097)	0.753(0.0064)	0.291 (0.0086)	0.390 (0.0014)	0.369 (0.0132)	0.333 (0.0005)	OOT	OOT
DMGI	0.473 (0.0155)	0.626 (0.0093)	0.220 (0.0214)	0.392 (0.0026)	0.548 (0.0190)	0.544 (0.0189)	0.781 (0.0303)	0.787 (0.0235)
FAME	0.722 (0.0114)	0.727 (0.0091)	0.323 (0.0154)	0.393 (0.0060)	0.593 (0.0135)	0.594 (0.0143)	0.842 (0.0183)	0.868 (0.0127)
DualHGNN	/	/	0.347 (0.0114)	0.402 (0.0127)	/	/	/	/
SGC	0.516 (0.0047)	0.587 (0.0157)	0.286 (0.0231)	0.361 (0.0175)	0.489 (0.0106)	0.563 (0.0133)	0.622 (0.0009)	0.623 (0.0009)
AM-GCN	0.702 (0.0175)	0.713 (0.0223)	0.307 (0.0232)	0.399 (0.0156)	0.610 (0.0021)	0.640 (0.0013)	0.867 (0.0105)	0.878 (0.0112)
R-GCN	0.690 (0.0078)	0.692 (0.0106)	0.265 (0.0326)	0.381 (0.0125)	0.544 (0.0172)	0.572 (0.0145)	0.862 (0.0053)	0.870 (0.0070)
HAN	0.690 (0.0149)	0.726 (0.0086)	0.275 (0.0327)	0.392 (0.0081)	0.552 (0.0112)	0.568 (0.0078)	0.806 (0.0078)	0.813 (0.0100)
NARS	0.722(0.0103)	0.721(0.0097)	0.297 (0.0201)	0.392 (0.0195)	0.565 (0.0037)	0.574 (0.0048)	0.794 (0.0255)	0.804 (0.0320)
MAGNN	0.755 (0.0105)	0.757 (0.0133)	0.348 (0.0488)	0.398 (0.0405)	0.614 (0.0073)	0.615 (0.0089)	0.881 (0.0284)	0.895 (0.0396)
HPN	0.710(0.0612)	0.732(0.0490)	0.263 (0.0346)	0.392 (0.0405)	0.578 (0.0023)	0.584 (0.0021)	0.822 (0.0201)	0.830 (0.0201)
GTN	0.749(0.0124)	0.751(0.0132)	0.255 (0.0420)	0.392 (0.0071)	0.615 (0.0108)	0.616 (0.0093)	0.852 (0.0137)	0.868 (0.0125)
HGSL	0.754 (0.0100)	0.758 (0.0103)	0.338 (0.0121)	0.398 (0.0238)	0.620 (0.0048)	0.638 (0.0030)	0.893 (0.0284)	0.902 (0.0396)
HGTN	0.765(0.0057)	0.738(0.0129)	0.331 (0.0213)	0.387 (0.0156)	0.617 (0.0087)	0.628 (0.0106)	0.825 (0.0098)	0.832 (0.0104)
MHGCN	0.868 (0.0186)	0.875 (0.0152)	0.351 (0.0204)	0.458 (0.0160)	0.764 (0.0145)	0.782 (0.0138)	0.945 (0.0221)	0.952 (0.0203)
MHGCN+	**0.903 (0.0137)**	**0.908 (0.0142)**	**0.369 (0.0198)**	**0.461 (0.0146)**	**0.784 (0.0192)**	**0.796 (0.0144)**	**0.956 (0.0124)**	**0.960 (0.0123)**

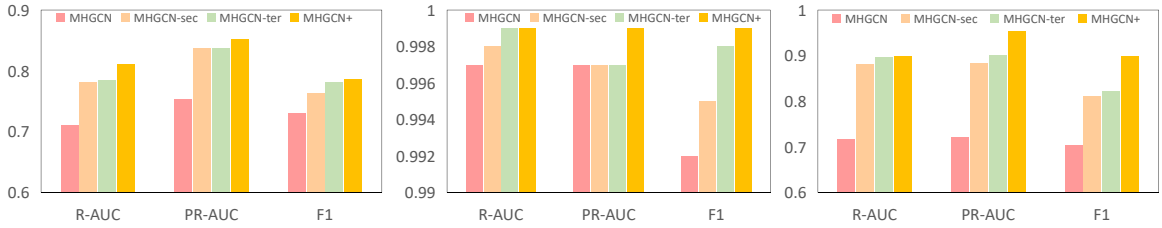
OOT: Out Of Time (36 hours), OOM: Out Of Memory. The standard deviations are reported in parentheses.

From the experimental results, we again see that our extended MHGCN+ improves the performance of the previous MHGCN for node classification. Specifically, MHGCN+ performs better than MHGCN by an average of 3.24% and 1.76% in terms of Macro-F1 and Micro-F1 across all tested datasets. The reason for this improvement is that MHGCN+ overcomes the shortcoming of importance conflict of meta-paths in MHGCN, so it can obtain more appropriate weights for different meta-paths.

(a) Macro-F1 score

(b) Micro-F1 score

Fig. 5. Experimental results of ablation study



(a) AMiner (b) Alibaba (c) DBLP

Fig. 6. Link prediction performance of four multiplex relation aggregations on AMiner, Alibaba, and DBLP

7.6 Ablation Study

To validate the effectiveness of each component of our model, we further conduct experiments on different model variations. Here MHGCN-R does not consider the importance of different relations, that is, we set the weights β_r to 1; MHGCN-L uses only a two-layer GCN to obtain the embedding, so it can only capture the 2-length meta-paths. We report the results of the ablation study on four datasets for node classification in Figure 5, where the performance on Alibaba refers to the right-ordinate axis.

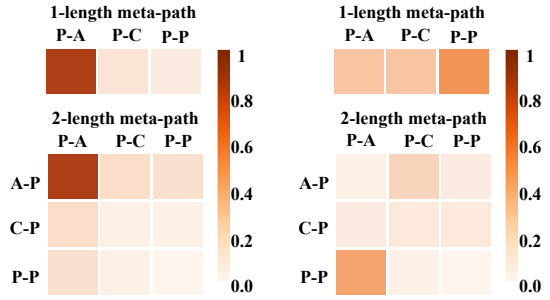
It can be seen from the results that the two key components both contribute to the performance improvement of our MHGCN. The comparison between MHGCN-R and MHGCN highlights the effectiveness of the importance of different relations. We can observe that MHGCN-R performs worse than MHGCN on all datasets in terms of both Macro-F1 and Micro-F1 metrics, reducing 9.68% performance in Macro-F1 score on Alibaba, which demonstrates the crucial role of our designed multiplex relation aggregation module in capturing the importance of different relations for node representation learning. The comparison between MHGCN-L and MHGCN reflects the importance of our multilayer graph convolution module. Compared with MHGCN-L, MHGCN improves 2.97%, 18.98%, 4.09%, and 1.51% over MHGCN-L in terms of Macro-F1 on AMiner-s, Alibaba, IMDB, and DBLP, respectively. This indicates that our proposed multilayer graph convolution module effectively captures useful meta-paths of different lengths across multiplex relations.

To evaluate the effectiveness of the proposed four multiplex relation aggregations, we further perform experiments on MHGCN+ variations for link prediction. We report the experimental results on three datasets in Figure 6, where MHGCN-sec and MHGCN-ter adopt the secondary independent aggregation and tertiary independent aggregation in multiplex relation aggregation, respectively.

From the results in Figure 6, we can see that MHGCN+ achieves the best performance, while MHGCN performs the worst performance in terms of all evaluation metrics on three datasets. MHGCN-ter is slightly better than MHGCN-sec in most cases. In particular, MHGCN+ outperforms MHGCN by 25.21%, 32.13%, and 27.88% in terms of ROC-AUC, PR-AUC, and F1 score on DBLP network, respectively. The comparison between these variants adequately demonstrates the effectiveness of our proposed four independent aggregation methods in learning the importance of different meta-paths and the superiority of the fully independent aggregation adopted by our MHGCN+.

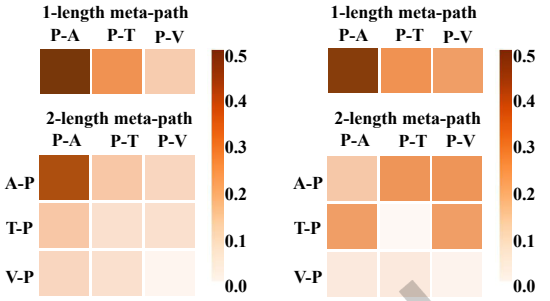
7.7 Visualization of Learned Relation Weights

To further illustrate the difference between MHGCN+ and MHGCN, we visualize the learned relation (or meta-path) weights in the link prediction task on AMiner and DBLP datasets. Experimental results are depicted in Figure 7 and Figure 8, where P, A, C, T and V denote Paper, Author, Conference, Term, and Venue, respectively.



(a) MHGCN (b) MHGCN+

Fig. 7. Visualization of relation weights on AMiner



(a) MHGCN (b) MHGCN+

Fig. 8. Visualization of relation weights on DBLP

Since MHGCN uses primary independent aggregation, *i.e.*, each aggregation shares the same set of parameters $\{\beta_r | r = 1, 2, \ldots, |\mathcal{R}|\}$. Therefore, MHGCN only learns the weight of each relation, and the weights of length-2 meta-paths are determined by $\{\beta_r | r = 1, 2, \ldots, |\mathcal{R}|\}$. As shown in Figure 7(a) and Figure 8(a), the weights of length-2 meta-paths are related to $\{\beta_r | r = 1, 2, \ldots, |\mathcal{R}|\}$ and are symmetric. For example, on AMiner, the weights of P-A, P-C, and P-P relations have determined the weights of nine length-2 meta-paths (*e.g.*, A-P-A, A-P-C, A-P-P, C-P-C, C-P-P, P-P-P). MHGCN+ uses fully independent aggregation, *i.e.*, each type of meta-path is associated with an independent weight. Therefore, MHGCN+ can capture the importance of different types of meta-paths, including those of different lengths. As shown in Figure 8, MHGCN+ learns similar weights as MHGCN for different relations (*i.e.*, P-A, P-T, and P-V), but for length-2 meta-paths, MHGCN+ can learn independent weights in link prediction task independent of the weights of the different relations. This is also the main reason why our extended MHGCN+ further outperforms MHGCN on link prediction.

7.8 Model Efficiency Analysis

MHGCN mainly consists of two computing modules: Multiplex Relation Aggregation Module and Multilayer Graph Convolution Module. The time complexity of Multiplex Relation Aggregation Module is $O(|\mathcal{R}|n^2)$, and the time complexity of graph convolution is $O(n^2dl + nmd + nd^2(l-1))$, so the total time complexity of MHGCN is $O(n^2(|\mathcal{R}| + dl) + nmd + nd^2(l-1))$.

MHGCN+ is also divided into two modules, the Multiplex Relation Aggregation Module and the Multilayer Graph Convolution Module. The time complexity of the Multiplex Relation Aggregation Module is $O(|\mathcal{R}|^l n^2)$, and the time complexity of the Multiplex Relation Aggregation Module is $O(n^2dl + nmd + nd^2(l-1))$, the total time complexity of MHGCN is $O(n^2(|\mathcal{R}|^l + dl) + nmd + nd^2(l-1))$. It can be seen that MHGCN and MHGCN+ are in the same order of magnitude in terms of time complexity.

For GTN, the adjacency matrix needs to be accumulated, and the time complexity is $O(n^3)$, the complexity of this process is too high, making it far less computationally efficient than our method.

We also compare the efficiency of our model with other GNN baselines for semi-supervised node classification. We report the experimental results on four datasets in Table 5.

As can be seen from Table 5, our MHGCN achieves the fourth-best performance after three heterogeneous network embedding methods (*i.e.*, R-GCN, NARS and HPN). However, from the above experimental results (Tables 3 and 4), MHGCN is significantly better than these three methods in both link prediction and node classification. MHGCN is significantly faster than the best performed GNN baseline in node classification task (*i.e.*, HGSL) on all datasets under the same number of training rounds. More specifically, our MHGCN achieves up

Table 5. Runtime comparison of GNN methods (Second)

Method	AMiner-M	Alibaba	IMDB	DBLP
AM-GCN	8703.71	2519.82	24280.12	2786.73
R-GCN	**153.04**	301.25	155.40	192.85
HAN	87105.55	4226.95	70510	22315.36
NARS	172.21	**211.54**	**75.81**	**108.54**
MAGNN	10361.20	2320.62	731.03	2125.33
HPN	172.82	249.47	176.64	109.49
GTN	OOM	21166.83	4287.20	18233.64
HGSL	1684.03	2120.93	1758.21	2037.10
DualHGN	/	11295.92	/	/
MHGCN	645.20	996.52	677.23	970.29
Speedup*	**135.05**×	**4.37**×	**104.15**×	**23.01**×
Speedup**	/	**21.25**×	**6.33**×	**18.80**×
MHGCN+	710.64	1675.01	1045.08	5149.10
Speedup+	**122.57**×	**2.52**×	**67.47**×	**4.33**×
Speedup++	/	**12.64**×	**4.10**×	**3.54**×

* Speedup of MHGCN over HAN.
** Speedup of MHGCN over GTN.
+ Speedup of MHGCN+ over HAN.
++ Speedup of MHGCN+ over GTN.
OOM: Out Of Memory.

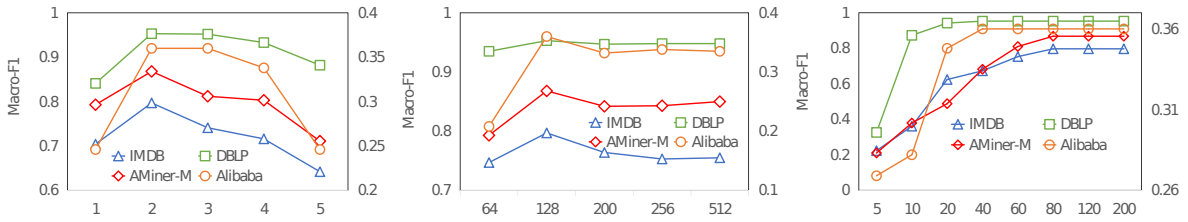
to 135× speedup over state-of-the-art embedding method HAN. MHGCN is faster than state-of-the-art AMHEN embedding method GTN by 21.25 times on multiplex Alibaba network. MHGCN is even 2.33 times and 16.58 times faster than state-of-the-art heterogeneous GNN model MAGNN on Alibaba and AMiner-M, respectively. The main reason is because our MHGCN adopts the idea of simplifying graph convolutional networks, that is, omitting non-linear activation function. Therefore, the training efficiency of MHGCN can be significantly improved.

Due to the introduction of more parameters in MHGCN+, the training process consumes more time compared to MHGCN. According to Table 5, it can be seen that the time consumed by MHGCN+ is on the same order of magnitude as MHGCN, but there is a significant improvement in node classification. Moreover, our new MHGCN+ still shows significant speed improvement compared to other baseline methods such as HAN and GTN. For example, our MHGCN+ still achieves up to 122× speedup over state-of-the-art embedding method HAN.

In fact, according to the above experimental results in Figure 9(c), our model can converge quickly within 80 rounds for node classification on four tested datasets, that is, our model does not need to be trained for 200 rounds set in our experimental evaluation and thus can achieve faster efficiency.

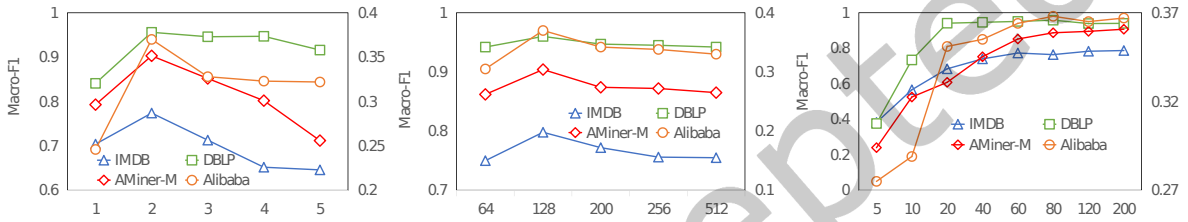
7.9 Parameter Sensitivity

We finally investigate the parameter sensitivity of our MHGCN and MHGCN+ with respect to the number of layers l (*i.e.*, the length of meta-paths l for MHGCN+), embedding dimension d, and the number of training rounds. We show the Macro-F1 score on the node classification task with different parameter settings on four datasets in Figure 9 and Figure 10. Notice that the performance on Alibaba refers to the right-ordinate axis, and



(a) Macro-F1 score *w.r.t.* #layers (b) Macro-F1 score *w.r.t.* dimension d (c) Macro-F1 score *w.r.t.* #rounds

Fig. 9. Parameter sensitivity of proposed MHGCN *w.r.t.* #layers, dimension d, and #rounds.



(a) Macro-F1 score *w.r.t.* #lengths (b) Macro-F1 score *w.r.t.* dimension d (c) Macro-F1 score *w.r.t.* #rounds

Fig. 10. Parameter sensitivity of proposed MHGCN+ *w.r.t.* #lengths, dimension d, and #rounds.

the number of training rounds on AMiner-M dataset is multiplied by five and three for MHGCN and MHGCN+, respectively.

As we see, the extended MHGCN+ exhibits the same performance trend as MHGCN with respect to the three parameters on four datasets. From the results in Figure 9(a) and Figure 10(a), we can observe that the performance of MHGCN and MHGCN+ first increases as l increases, and then the performance begins to decline when $l \geq 2$. This is mainly because 1-length and 2-length meta-path interactions already effectively capture the topological structures of networks for node classification, while longer meta-paths would not lead to performance gains. As the number of GCN layers grows, the representation of nodes would be flattened after multiple convolutions, resulting in performance degradation.

Then we analyze the impact of embedding dimension on model performance. As shown in Figure 9(b) and Figure 10(b), the performance of MHGCN and MHGCN+ first gradually rises and then decreases slightly as dimension d increases, and achieves the best performance when embedding dimension $d = 128$ on the four datasets. This is because when the dimension d is small, the features of all nodes are compressed into a small embedding space, so it is difficult to preserve the feature proximities of all node pairs. On the contrary, a larger dimension would also flatten the distance between all node embeddings, which fails to reflect the proximity between nodes.

Figure 9(c) and Figure 10(c) illustrate the performance of our MHGCN and MHGCN+ with respect to the number of training rounds for model learning. We can conclude that our methods can converge quickly, and efficiently to achieve stable performance within 80 rounds on almost all test datasets, which reflects the high efficiency of our model.

8 CONCLUSION

In this paper, we propose a novel graph neural network model MHGCN+ for attributed multiple heterogeneous network embedding. Our model mainly consists of two key components: multiplex relation aggregation and multilayer graph convolution module. Through multiple relation aggregation, MHGCN+ can distinguish the importance of different relations in different meta-paths in multiplex heterogeneous networks. With the multilayer graph convolution module, MHGCN+ can automatically capture short and long meta-path interactions across multiple relations, and learn meaningful node embeddings through model parameter learning during the training phase. Experimental results on seven real-world datasets demonstrate the superiority of the proposed MHGCN+ in both link prediction and node classification. As future work, we will focus on investigating more efficient graph neural networks for large-scale heterogeneous network embedding.

ACKNOWLEDGMENTS

This work is partially supported by the National Natural Science Foundation of China under grant Nos. 62176243, 61773331, and 41927805, and the National Key Research and Development Program of China under grant Nos. 2018AAA0100602 and 2019YFC1509100.

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2623–2631.

[2] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. In *KDD*. 1358–1368.

[3] Haochen Chen, Syed Fahad Sultan, Yingtao Tian, Muhao Chen, and Steven Skiena. 2019. Fast and Accurate Network Embeddings via Very Sparse Random Projection. In *CIKM*. 399–408.

[4] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. 2018. PME: projected metric embedding on heterogeneous networks for link prediction. In *KDD*. 1177–1186.

[5] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*. 135–144.

[6] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *CIKM*. 1797–1806.

[7] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *WWW*. 2331–2341.

[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 855–864.

[9] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 1025–1035.

[10] Li He, Hongxu Chen, Dingxian Wang, Shoaib Jameel, Philip Yu, and Guandong Xu. 2021. Click-Through Rate Prediction with Multi-Modal Hypergraphs. In *CIKM*. 690–699.

[11] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*. 639–648.

[12] Binbin Hu, Yuan Fang, and Chuan Shi. 2019. Adversarial Learning on Heterogeneous Information Networks. In *KDD*. 120–129.

[13] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *WWW*. 2704–2710.

[14] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xia, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. 2021. Knowledge-aware coupled graph neural network for social recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4115–4122.

[15] Houye Ji, Xiao Wang, Chuan Shi, Bai Wang, and S Yu Philip. 2021. Heterogeneous graph propagation network. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2021), 521–532.

[16] Baoyu Jing, Chanyoung Park, and Hanghang Tong. 2021. Hdmi: High-order deep multiplex infomax. In *The Web Conference*. 2414–2424.

[17] Thomas Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *ArXiv* abs/1609.02907 (2016). https://api.semanticscholar.org/CorpusID:3144218

[18] Mengran Li, Yong Zhang, Xiaoyong Li, Yuchen Zhang, and Baocai Yin. 2023. Hypergraph transformer neural networks. *ACM Transactions on Knowledge Discovery from Data* 17, 5 (2023), 1–22.

[19] Weiyi Liu, Pin-Yu Chen, Sailung Yeung, Toyotaro Suzumura, and Lingli Chen. 2017. Principled multilayer network embedding. In *ICDMW*. IEEE, 134–141.
[20] Zhijun Liu, Chao Huang, Yanwei Yu, and Junyu Dong. 2021. Motif-preserving dynamic attributed network embedding. In *WWW*. 1629–1638.
[21] Zhijun Liu, Chao Huang, Yanwei Yu, Baode Fan, and Junyu Dong. 2020. Fast Attributed Multiplex Heterogeneous Network Embedding. In *CIKM*. 995–1004.
[22] Zhijun Liu, Chao Huang, Yanwei Yu, Peng Song, Baode Fan, and Junyu Dong. 2020. Dynamic representation learning for large-scale attributed networks. In *CIKM*. 1005–1014.
[23] Xiaoling Long, Chao Huang, Yong Xu, Huance Xu, Peng Dai, Lianghao Xia, and Liefeng Bo. 2021. Social Recommendation with Self-Supervised Metagraph Informax Network. In *CIKM*. 1160–1169.
[24] Yuanfu Lu, Chuan Shi, Linmei Hu, and Zhiyuan Liu. 2019. Relation structure-aware heterogeneous information network embedding. In *AAAI*. 4456–4463.
[25] Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. 2020. Unsupervised Attributed Multiplex Network Embedding. In *AAAI*. 5371–5378.
[26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *KDD*. 701–710.
[27] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. 2019. Netsmf: Large-scale network embedding as sparse matrix factorization. In *WWW*. 1509–1520.
[28] 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 *European semantic web conference*. Springer, 593–607.
[29] Jingbo Shang, Meng Qu, Jialu Liu, Lance M Kaplan, Jiawei Han, and Jian Peng. 2016. Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. *arXiv preprint arXiv:1610.09769* (2016).
[30] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. 2018. Heterogeneous information network embedding for recommendation. *TKDE* 31, 2 (2018), 357–370.
[31] Yu Shi, Huan Gui, Qi Zhu, Lance M. Kaplan, and Jiawei Han. 2018. AspEm: Embedding Learning by Aspects in Heterogeneous Information Networks. In *SDM*. 144–152.
[32] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. 1067–1077.
[33] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. 2017. Graph attention networks. *stat* 1050, 20 (2017), 10–48550.
[34] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. ACM, 2022–2032.
[35] Xiao Wang, Yuanfu Lu, Chuan Shi, Ruijia Wang, Peng Cui, and Shuai Mou. 2020. Dynamic heterogeneous information network embedding with meta-path based proximity. *IEEE Transactions on Knowledge and Data Engineering* 34, 3 (2020), 1117–1132.
[36] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020. Am-gcn: Adaptive multi-channel graph convolutional networks. In *KDD*. 1243–1253.
[37] Yueyang Wang, Ziheng Duan, Binbing Liao, Fei Wu, and Yueting Zhuang. 2019. Heterogeneous Attributed Network Embedding with Graph Convolutional Networks. In *AAAI*. 10061–10062.
[38] Wei Wei, Chao Huang, Lianghao Xia, Yong Xu, Jiashu Zhao, and Dawei Yin. 2022. Contrastive meta learning with behavior multiplicity for recommendation. In *KDD*. 1120–1128.
[39] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, et al. 2019. Simplifying Graph Convolutional Networks. In *ICML*. 6861–6871.
[40] Yongji Wu, Defu Lian, Yiheng Xu, Le Wu, and Enhong Chen. 2020. Graph convolutional networks with markov random field reasoning for social spammer detection. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 1054–1061.
[41] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Bo Zhang, and Liefeng Bo. 2020. Multiplex Behavioral Relation Learning for Recommendation via Memory Augmented Transformer Network. In *SIGIR*. 2397–2406.
[42] Lianghao Xia, Yong Xu, Chao Huang, Peng Dai, and Liefeng Bo. 2021. Graph meta network for multi-behavior recommendation. In *SIGIR*. 757–766.
[43] Hansheng Xue, Luwei Yang, Vaibhav Rajan, Wen Jiang, Yi Wei, and Yu Lin. 2021. Multiplex bipartite network embedding using dual hypergraph convolutional networks. In *WWW*. 1649–1660.
[44] Yuhao Yang, Chao Huang, Lianghao Xia, Yuxuan Liang, Yanwei Yu, and Chenliang Li. 2022. Multi-behavior hypergraph-enhanced transformer for sequential recommendation. In *KDD*. 2263–2274.
[45] Lingfan Yu, Jiajun Shen, Jinyang Li, and Adam Lerer. 2020. Scalable Graph Neural Networks for Heterogeneous Graphs. *CoRR* abs/2011.09679 (2020). arXiv:2011.09679 https://arxiv.org/abs/2011.09679
[46] Pengyang Yu, Chaofan Fu, Yanwei Yu, Chao Huang, Zhongying Zhao, and Junyu Dong. 2022. Multiplex Heterogeneous Graph Convolutional Network. In *KDD*. 2377–2387.

[47] Yanwei Yu, Huaxiu Yao, Hongjian Wang, Xianfeng Tang, and Zhenhui Li. 2018. Representation learning for large-scale dynamic networks. In *DASFAA*. Springer, 526–541.
[48] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. 2019. Graph Transformer Networks. In *NeurIPS*. 11960–11970.
[49] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *KDD*. 793–803.
[50] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. 2018. Scalable multiplex network embedding. In *IJCAI*, Vol. 18. 3082–3088.
[51] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. 2019. STAR-GCN: stacked and reconstructed graph convolutional networks for recommender systems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (Macao, China) *(IJCAI'19)*. AAAI Press, 4264–4270.
[52] Ziwei Zhang, Peng Cui, Haoyang Li, Xiao Wang, and Wenwu Zhu. 2018. Billion-Scale Network Embedding with Iterative Random Projection. In *ICDM*. 787–796.
[53] Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. 2021. Heterogeneous graph structure learning for graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4697–4705.