Data Mining and Knowledge Discovery (2023) 37:1518–1547
https://doi.org/10.1007/s10618-023-00932-w

# A graph convolutional fusion model for community detection in multiplex networks

**Xiang Cai**[1] · **Bang Wang**[1]

Received: 25 August 2022 / Accepted: 28 February 2023 / Published online: 6 April 2023


## Abstract

Community detection is to partition a network into several components, each of which contains densely connected nodes with some structural similarities. Recently, multiplex networks, each layer consisting of a same node set but with a different topology by a unique edge type, have been proposed to model real-world multi-relational networks. Although some heuristic algorithms have been extended into multiplex networks, little work on neural models have been done so far. In this paper, we propose a graph convolutional fusion model (GCFM) for community detection in multiplex networks, which takes account of both intra-layer structural and inter-layer relational information for learning node representation in an interwoven fashion. In particular, we first develop a graph convolutional auto-encoder for each network layer to encode neighbor-aware intra-layer structural features under different convolution scales. We next design a multiscale fusion network to learn a holistic version of nodes' representations by fusing nodes' encodings at different layers and different scales. Finally, a self-training mechanism is used to train our model and output community divisions. Experiment results on both synthetic and real-world datasets indicate that the proposed GCFM outperforms the state-of-the-art techniques in terms of better detection performances.

**Keywords** Community detection · Multiplex network · Graph neural networks · Deep learning

Responsible editor: Jingrui He

✉ Bang Wang
wangbang@hust.edu.cn

Xiang Cai
cxiang@hust.edu.cn

[1] School of Electronic Information and Communication, Huazhong University of Science and Technology, Wuhan 430074, Wuhan, China

# 1 Introduction

## 1.1 Background

Community detection is to partition a network into several components, each of which, so-called a community, contains densely connected nodes with some structural similarities (Fortunato 2010; Bouguessa et al. 2010). As an important task in network science, community detection can find many real-world applications in diverse domains, like biology, chemistry, transportation, sociology (Garcia et al. 2018; Chang et al. 2016; Liu and Wang 2022; Magalingam et al. 2015). For example, community detection in biological networks can be used to analyze the interaction of brain regions and its influence on brain functions (Garcia et al. 2018). Community detection in social networks can help identifying crime organizations (Magalingam et al. 2015).

For its wide applications, many algorithms have been developed for community detection, including traditional heuristic algorithms and modern deep learning ones. Those heuristic algorithms, such as GN (Newman and Girvan 2004) and Louvain Blondel et al. (2008), usually optimize an objective function iteratively to improve the quality of detected communities. Modern deep learning-based algorithms, such as CommunityGAN (Jia et al. 2019), DANMF (Ye et al. 2018), and MRFasGCN (Jin et al. 2019), encode graph structural information for learning nodes' representations that are used to compute community divisions.

Although lots of progresses have been achieved in the field (Souravlas et al. 2021; Su et al. 2022), most algorithms for community detection are developed in the context of single layer networks. A single layer network, or called a *monoplex network* in this paper, can be characterized by its unique type of edge connecting two nodes. However, in many practical scenarios multiple types of relations do exist in between two entities. For example, people can establish either friend relation or coworker relation, or both in a social network. For better representing different relations, it is recently proposed to use a *multiplex network* consisting of multiple layers of networks each representing only one relation type. Multiplex networks can be applied to lots of applications such as multi-behavior recommendation (Xia et al. 2021) and community detection (Tang et al. 2009; Magnani et al. 2021).

A multiplex network is composed of $L$ network layers, $G_l(V, E_l)$, $l = 1, 2, ..., L$. Figure 1 illustrates a 3-layer network for community detection. The number of nodes in each layer are the same, denoted as $N$. A multiplex network can be also represented as multiple adjacency matrices, $\mathbf{A}_l \in \mathbb{R}^{N \times N}$, for $l = 1, 2, ..., L$. Similar to monoplex networks, community detection in multiplex networks aims to partition the node set $V$ into $M$ communities, $\{C_1, C_2, ..., C_M\}$. The detected communities can be seen as the comprehensive associations of nodes under diverse types of relations.

In recent years, some solutions have been proposed to address community detection in multiplex networks. A few of them propose to first convert this problem into the classical setting of community detection in a single layer network (Berlingerio et al. 2011; Suthers et al. 2013; Shao et al. 2022). For example, Berlingerio et al. (2011) first reduce a multiplex network into a single layer one by judging whether the nodes are connected in at least one network layer and then apply the random walk method

to detect communities from this single layer network. Some algorithms directly apply a greedy modularity-maximization strategy on a multiplex network (Mucha et al. 2010; Tagarelli et al. 2017; Pramanik et al. 2017; Paul and Chen 2022). For example, Tagarelli et al. (2017) propose a multilayer modularity function to find consensus community structures by a greedy search approach. Some other algorithms apply matrix decomposition to extract features from a multiplex network (Ma et al. 2018; Gligorijević et al. 2019; Chen et al. 2019). For example, Ma et al. (2018) propose a nonnegative matrix factorization (NMF) algorithm called s2-jNMF, which uses a joint NMF for each network layer to obtain multiplex basis matrix for community discovering.

### 1.2 Motivation

The limitation of existing approaches on community detection in multiplex networks mainly lies in the following two aspects. On the one hand, lots of algorithms (Berlingerio et al. 2011; Boutemine and Bouguessa 2017; Pramanik et al. 2017; Interdonato et al. 2017) have considered how to extend traditional heuristic ones in single layer networks for multiplex networks, which could lead to the loss of some inter-layer relational information. For example, Boutemine and Bouguessa (2017) flatten a multiplex network into a single one and then utilize the Label Propagation Algorithm (LPA) (Raghavan et al. 2007). However, a flattened edge cannot describe all latent relations between two nodes in a multiplex network. On the other hand, some representation learning algorithms used in multiplex networks (Park et al. 2020; Jing et al. 2021) can be applied for community detection as their downstream tasks, but they ignore the critical factors of building multilayer community structures. For example, Jing et al. (2021) propose to maximize the mutual information between the local node embedding and the global summary for node classification and community detection. But they mainly focus on how to train a general node embedding. Besides, they are normally based on the prior knowledge of nodes' attributes for representation learning. How to detect communities from only topological structures of a multiplex network without extra node information is a challenge task for most of such node representation learning models.

To overcome such limitations, we consider the following two challenging issues: (a) What kind of structural or topological characteristics are suitable for a community? (b) How to extract and fuse features from each single layer to represent a community in a multiplex network? We note that two kinds of characteristics are important for a community, i.e. path-aware topological characteristics and neighbor-aware structural characteristics. The former can be learned by sampling short node sequences via random walk and maximizing nodes' co-occurrence probability in sequences. The latter can be learned by merging the embedding of a node from its neighbors (Liu et al. 2022). Besides, neighbors of different scales can be used for embedding learning. We also note that intra-layer and inter-layer not only can be extracted but also can be fused to learn nodes' representations for community detection via neural network models.

Motivated from such considerations, this paper considers to learn not only neighbor-aware intra-layer structural information but also semantics-aware inter-layer relational

information to learn nodes' representation for community detection. In many cases, little prior knowledge is available about which type of information. For example, each layer of the temporal multiplex network represents a time slice and cannot provide extra information from the semantic perspective. Therefore, intra-layer or inter-layer, is more important for community detection in multiplex networks. We further argue that the two kinds of information should be jointly exploited for node representation learning in an interwoven manner, rather than in a separate way. In response to these arguments, we design a *graph convolutional fusion model* (GCFM) for community detection in multiplex networks.

### 1.3 Contribution

The novelties of our proposed GCFM framework include encoding and fusing intra-layer structural information with inter-layer relational information for learning node representation in an interwoven fashion. Specifically, there are three key contributions in our GCFM: (1) How to learn node topological characteristics in each single layer? In the GCFM, we first employ some graph embedding technique to obtain node initial embedding for each network layer and next design a module of *graph convolutional auto-encoder* (GCA), executed on a per network layer basis, to encode neighbor-aware intra-layer structural information under different convolution scales .[1] (2) How to better learn inter-layer semantics for community detection? In the GCFM, we design a module of *multiscale fusion network* (MFN) to fuse nodes' encodings at different layers and different scales for learning a holistic version of nodes' representations. (3) How to detect community based on the node feature space? In the GCFM, we use a self-training mechanism to train our model and output community divisions for a multiplex network.

Compared with those traditional heuristic studies (Tang et al. 2009; Boutemine and Bouguessa 2017; Gligorijević et al. 2019), our GCFM uses the *random walk* to generate node initial embedding for subsequent encoding and fusing, instead of optimizing a certain indicator such as modularity, which avoids falling into local optimum. For those deep learning-based methods which only use auto-encoder (Song and Thiagarajan 2019), our GCFM designs the GCA and MFN that consider the influence of different convolutional scales. For other node embedding based methods which intend to train a general node embedding for downstream tasks such as node classification, community detection etc. (Park et al. 2020; Jing et al. 2021), our GCFM applies a community specific training loss and pays attention to the community characteristics when learning nodes' representations. Extensive experiments are conducted on both synthetic and real-world datasets. Results validate that our GCFM achieves higher precision and detects communities with better modularity over peer competitors in most cases.

The main contributions are summarized as follows:

- Propose a graph convolutional fusion model (GCFM) to encode and fuse intra-layer and inter-layer information for representation learning in an interwoven fashion for community detection in multiplex networks.

[1] In order to distinguish the concept of "network layer", we use "scale" in the context of graph convolutional networks.

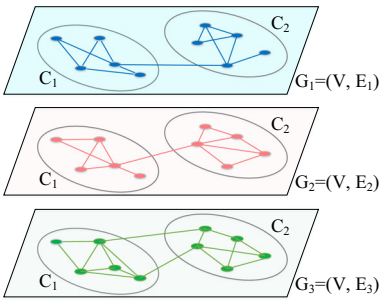



**Fig. 1** A 3-layer network with two communities

- Develop a graph convolutional auto-encoder (GCA) to encode neighbor-aware intra-layer structural information at different scales.
- Design a multiscale fusion network (MFN) to fuse and encode intra-layer structural information and inter-layer relational information at different layers and different scales.
- Experiment on both synthetic and real-world datasets to validate the superiority of our proposed model.

The rest of the paper is organized as follows. Section 2 reviews the related work. The proposed GCFM is presented in Sect. 3 and evaluted in Sect. 4 and 5. Section 6 concludes the paper.

## 2 Related work

### 2.1 Multiplex network community detection

Many traditional heuristic algorithms for community detection in monoplex networks have been extended into multiplex ones, which can be generally categorized into three types: flattening, aggregation, and direct methods (Huang et al. 2021).

Flattening methods (Berlingerio et al. 2011; Gao et al. 2019) first convert (flatten) a multiplex network into a monoplex one by, say for example, setting new edge weights and then use any monoplex algorithm to find communities. For example, Gao et al. (2019) propose a modified particle competition model, which constructs a extended adjacency matrix to represent a multiplex network for community detection.

Aggregation methods (Tang et al. 2009; Tagarelli et al. 2017; Ali et al. 2019) capture structural information from each single layer and aggregate them into a comprehensive version of node feature for community detection. For example, Ali et al. (2019) utilize a variational Bayes method to extract community structures from each network layer and aggregate them to determine the final community divisions.

Direct methods (Ma et al. 2018; Chen et al. 2019; Mercorio et al. 2019) work directly on a multiplex network to detect communities. For example, Ma et al. (2018) propose a s2-jNMF algorithm, which simultaneously factorizes the matrices from each network layer to obtain community divisions.

### 2.2 Deep learning community detection

Many deep learning techniques have been applied for community detection in monoplex networks. Compared to most traditional methods, the advantage of deep learning lies in that it can automatically learn node representations via, say for example, encoding graph structural information (Liu et al. 2020).

Deep neural networks, like convolutional neural networks (Sperlí 2019), autoencoders (Cao et al. 2018), and generative adversarial networks (Jia et al. 2019), can help capturing structural relations in between nodes. For example, Jia et al. (2019) propose a GAN-based (Generative Adversarial Network) model to encode the membership strength of nodes to communities for community detection.

Deep graph embedding-based models convert nodes into a low-dimensional vector space to preserve graph structural information, such as deep non-negative matrix factorization (Ye et al. 2018), deep sparse filtering (Xie et al. 2018), and community embedding (Tu et al. 2018). For example, Ye et al. (2018) propose a DANMF model, which learns the hierarchical mappings between the original network and the community distribution by using a deep auto-encoder.

Graph neural network-based models can extract community structures from raw graph data by fusing graph mining and deep learning techniques (Jin et al. 2019; Wang et al. 2019; Bo et al. 2020). For example, Jin et al. (2019) propose an end-to-end model for semi-supervised community detection, which integrates a graph convolutional network with the Markov Random Field technique.

## 3 Graph convolutional fusion model

Figure 2 presents the framework of our graph convolutional fusion model (GCFM), which contains four modules: (1) node initial embedding, (2) graph convolutional auto-encoder (GCA), (3) multiscale fusion network (MFN), and (4) self-training community detection.

### 3.1 Node initial embedding

We employ some graph embedding technique to obtain node initial embedding per network layer. Since nodes are not with attributes in our problem setting, we choose the DeepWalk (Perozzi et al. 2014) to help capturing latent path-aware topological features for initializing a node embedding. We note that other techniques can also be used, and we will compare a few in our experiments.

For the $l$-th network layer, we use $\mathbf{X}_l \in \mathbb{R}^{N \times d_0}$ to denote the node initial embedding matrix obtained from the DeepWalk, where $d_0$ is the dimension of initial embedding. Although the initial embeddings can be directly used for community detection, such path-aware topological embeddings may not be able to well describe latent neighborhood information of a node, which, however, is often the key to the community detection task. To capture such local association characteristics, we next design the GCA module for encoding neighbor-aware structural information.

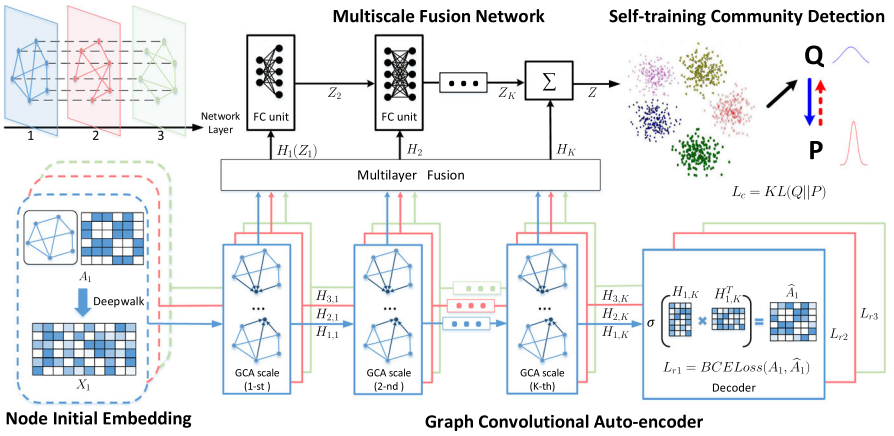



**Fig. 2** The overall framework of GCFM. $X_1$ is the node initial embedding learned, say for example, by the DeepWalk in the 1-st network layer. Different colors represent different network layers. The lower part shows a graph convolutional auto-encoder. The top part is a multiscale fusion network. Soft distribution $Q$ and target distribution $P$ indicate a self-training community detection mechanism

### 3.2 Graph convolutional auto-encoder

The GCA module executes aggregation operations to update a node's embedding from its neighbors', by which local structure information can be encoded into nodes' embeddings. Furthermore, by applying multiple scales of the aggregation operation, not only one-hop neighbors but also multi-hop neighbors as well their associate information through edges in between neighbors can be encoded for representing a node local structure at different scales.

The GCA aggregation operations are executed on each network layer independently. Take the $l$-th network layer for example. Let $\mathbf{A}_l \in \mathbb{R}^{N\times N}$ denote the adjacency matrix of the $l$-th network layer. For the first scale of GCA, the input includes $\mathbf{A}_l$ and $\mathbf{X}_l$, and the output is given by

$$\mathbf{H}_{l,1} = ReLU(\widetilde{\mathbf{D}}_l^{-\frac{1}{2}}\widetilde{\mathbf{A}}_l\widetilde{\mathbf{D}}_l^{-\frac{1}{2}}\mathbf{X}_l\mathbf{W}_{l,0}), \tag{1}$$

where $\widetilde{\mathbf{D}}_l^{-\frac{1}{2}}\widetilde{\mathbf{A}}_l\widetilde{\mathbf{D}}_l^{-\frac{1}{2}}$ is a normalized Laplacian matrix. $\widetilde{\mathbf{A}}_l = \mathbf{A}_l + \mathbf{I}$ is the adjacency matrix plus the node self-connection matrix and $\widetilde{\mathbf{D}}_{l(ii)} = \sum_j \widetilde{\mathbf{A}}_{l(ij)}$. $\mathbf{W}_{l,0} \in \mathbb{R}^{d_0\times d_1}$ is a learnable weight matrix. We choose the $ReLU(\cdot)$ function as the activation function.

The forward encoding process in the $k$-th GCA scale is as follows:

$$\mathbf{H}_{l,k} = ReLU(\widetilde{\mathbf{D}}_l^{-\frac{1}{2}}\widetilde{\mathbf{A}}_l\widetilde{\mathbf{D}}_l^{-\frac{1}{2}}\mathbf{H}_{l,k-1}\mathbf{W}_{l,k-1}), \tag{2}$$

where $\mathbf{H}_{l,k} \in \mathbb{R}^{N\times d_k}$ is the $l$-th layer node encoding after the $k$-th GCA scale. It is obtained from the node encoding $\mathbf{H}_{l,k-1} \in \mathbb{R}^{N\times d_{k-1}}$ of the previous scale. $\mathbf{W}_{l,k-1} \in \mathbb{R}^{d_{k-1}\times d_k}$ is a learnable weight matrix.

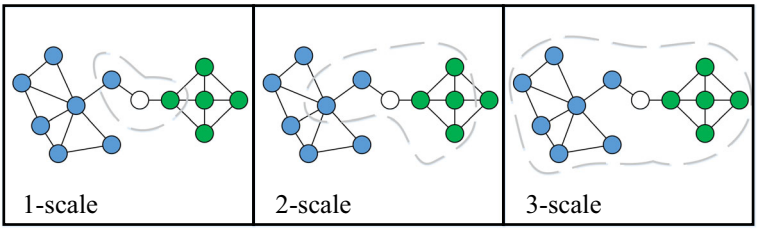



**Fig. 3** Illustration of using different scales of neighborhood to encode local structural information for the white node in a single network layer. Different numbers of neighbors, possibly from different communities, are included for updating its representation at different scales

The output of each GCA scale can be regarded as how large the neighborhood of a node will be included into its local structural encoding. Figure 3 illustrates how different GCA scales can include different neighborhoods of a node for encoding its local structure information, which could be analogy to encode a node-centric max-$k$-neighbor subgraph. The GCA module encodes multiscale local structures for each node; Such structural encodings can be exploited for discriminating neighborhood associations at different scales, which are key components for the community detection task.

In the decoding process, we reconstruct the edges in between nodes in each single layer by using an inner product decoder:

$$\widehat{\mathbf{A}}_l = sigmoid(\mathbf{H}_{l,K}\mathbf{H}_{l,K}^{\mathrm{T}}), \tag{3}$$

where $\widehat{\mathbf{A}}_l$ is the reconstructed adjacency matrix of the $l$-th layer. $\mathbf{H}_{l,K} \in \mathbb{R}^{N\times d}$ is the $K$-scale GCA node encoding.

We apply a binary cross entropy loss to minimize the difference between $\widehat{\mathbf{A}}_l$ and $\mathbf{A}_l$ for the $l$-th layer:

$$L_{rl} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}[\mathbf{A}_{l(ij)}\log(\widehat{\mathbf{A}}_{l(ij)}) + (1-\mathbf{A}_{l(ij)})\log(1-\widehat{\mathbf{A}}_{l(ij)})]. \tag{4}$$

The total reconstruction loss is the sum of reconstruction error of each layer:

$$L_r = \sum_{l}^{L} L_{rl}. \tag{5}$$

### 3.3 Multiscale fusion network

The GCA module is executed per network layer basis, that is, its node encodings at different neighborhood scales are for different network layers. As the edges in each layer express a kind of specific semantic relations among nodes, we need to take into account all kinds of semantic relations to output a comprehensive community

division for a multiplex network. To this end, we propose the MFN module to fuse node encodings not only from different GCA scales, but also from different network layers.

As we have no a prior knowledge to decide which layer in a multiplex network is more important, we first design the multilayer fusion of node structural encodings as a simple sum operation for the $k$-th GCA scale:

$$\mathbf{H}_k = \sum_{l=1}^{L} \mathbf{H}_{l,k}. \tag{6}$$

$\mathbf{H}_k \in \mathbb{R}^{N \times d_k}$ is called the $k$-th scale node multilayer encoding.

We next use a fully connected neural network to output different scales of node representation $\mathbf{Z}_k$. The initial scale of MFN fusion transforms $\mathbf{H}_1$ to the hidden state in the next scale:

$$\mathbf{Z}_1 = \mathbf{H}_1, \tag{7}$$

$$\mathbf{Z}_2 = ReLU(\mathbf{W}_1 \mathbf{Z}_1^{\mathrm{T}} + \mathbf{b}_1)^{\mathrm{T}}, \tag{8}$$

where $\mathbf{Z}_1 \in \mathbb{R}^{N \times d_1}$ is the initial scale of node representation, and $\mathbf{Z}_2 \in \mathbb{R}^{N \times d_2}$ the second scale of node representation. $\mathbf{W}_1 \in \mathbb{R}^{d_2 \times d_1}$ and $\mathbf{b}_1 \in \mathbb{R}^{d_2 \times N}$ are learnable parameters.

In the full connected units, each node exchanges its encodings with the other nodes'. We emphasize the topological information by adding the aggregated multilayer node encoding in each MFN scale. For the $k$-th ($k > 1$) MFN scale, we update the node representation $\mathbf{Z}_k$ by inputting the multilayer node encoding $\mathbf{H}_{k-1}$ and the multiscale node representation $\mathbf{Z}_{k-1}$ into a full connected unit. The forward process can be written as:

$$\mathbf{Z}_k = ReLU(\mathbf{W}_{k-1}(\mathbf{Z}_{k-1} + \mathbf{H}_{k-1})^{\mathrm{T}} + \mathbf{b}_{k-1})^{\mathrm{T}}, \tag{9}$$

where $\mathbf{Z}_k \in \mathbb{R}^{N \times d_k}$ and $\mathbf{Z}_{k-1}, \mathbf{H}_{k-1} \in \mathbb{R}^{N \times d_{k-1}}$. The weight matrix $\mathbf{W}_{k-1} \in \mathbb{R}^{d_k \times d_{k-1}}$ and bias $\mathbf{b}_{k-1} \in \mathbb{R}^{d_k \times N}$ are learnable parameters.

The final MFN scale is to fuse $\mathbf{Z}_K$ and $\mathbf{H}_K$ to output final node representation $\mathbf{Z} \in \mathbb{R}^{N \times d_K}$

$$\mathbf{Z} = \mathbf{Z}_K + \mathbf{H}_K, \tag{10}$$

which will be input to the next self-training module for community detection.

### 3.4 Self-training community detection

Community detection is often an unsupervised problem in the real world. Inspired by Xie et al. (2016), we adapt a self-training mechanism to train our GCFM.

For the $i$-th node, its final node representation $\mathbf{z}_i$, the $i$-th row of the node representation $\mathbf{Z}$, can be seen as an embedding in a feature space $\mathbb{Z}$. Let $\boldsymbol{v}_j$ denote the

representation of the $j$-th community center, which is also an embedding in the space $\mathbb{Z}$. All the community center embeddings are initialized by using the K-means algorithm before training.

Assume the pre-specified number of communities as $M$. The computation of community center is as follows. We first randomly select $M$ community centers. And then we repeat the following process until convergence: (a) For each node $i$, we find its class label $c_i$ which minimizes the distance between node $i$ and each community center $\boldsymbol{v}_j$, $j = 1, 2, ..., M$.

$$c_i = \arg\min_j (\|\mathbf{z}_i - \boldsymbol{v}_j\|^2). \tag{11}$$

(b) We next update each community center by recalculating the centroid of its belonging class.

$$\boldsymbol{v}_j = \frac{1}{|C_j|} \sum_{\mathbf{z}_i \in C_j} \mathbf{z}_i, \tag{12}$$

where $C_j$ includes all the node representation with class label $c_i = j$.

We next use the Student's t-distribution (Van der Maaten and Hinton 2008) as a kernel to measure the similarity between each node and each community center. The similarity $\mathbf{q}_{ij}$ between a node $i$ and a community $j$ is computed by

$$\mathbf{q}_{ij} = \frac{(1 + \|\mathbf{z}_i - \boldsymbol{v}_j\|^2)^{-1}}{\sum_{j'} (1 + \|\mathbf{z}_i - \boldsymbol{v}_{j'}\|^2)^{-1}}. \tag{13}$$

It can be interpreted as the probability of assigning node $i$ to community $j$. As such, we treat $\mathbf{Q} = [\mathbf{q}_{ij}] \in \mathbb{R}^{N \times M}$ as the distribution of the soft assignments of all nodes.

In order to detect cohesive communities, an objective is to enable each node closer to its belonging community center in terms of their similarity in the feature space $\mathbb{Z}$. This can be translated to obtain a high confidence and trustworthy distribution of soft assignments, denoted as the target distribution $\mathbf{P} \in \mathbb{R}^{N \times M}$, where an element $\mathbf{p}_{ij} \in \mathbf{P}$ can be computed by raising $\mathbf{q}_{ij}$ to the second power and then normalizing by frequency per community:

$$\mathbf{p}_{ij} = \frac{\mathbf{q}_{ij}^2 / \mathbf{f}_j}{\sum_{j'} \mathbf{q}_{ij'}^2 / \mathbf{f}_{j'}}, \tag{14}$$

where $\mathbf{f}_j = \sum_i \mathbf{q}_{ij}$ are soft community frequencies.

After acquiring soft assignment distribution $\mathbf{Q}$ and target distribution $\mathbf{P}$, we use the Kullback–Leibler divergence loss to measure the difference between the two community distributions.

$$L_c = KL(\mathbf{P} \| \mathbf{Q}) = \sum_i \sum_j \mathbf{p}_{ij} \log \frac{\mathbf{p}_{ij}}{\mathbf{q}_{ij}}. \tag{15}$$

By minimizing the KL divergence loss, our model can improve the distribution **Q** under the supervision of the target distribution **P**, which is called self-training community detection. Note that **P** should have the following properties: (a) a high accuracy of community division (b) an assigned community with high confidence (c) a normalized loss contribution of each centroid. Therefore, we choose **Q** to construct **P** because **Q** is natural and flexible for its use of softer probabilistic targets.

For model training, we define the final loss of GCFM by

$$L = L_r + \lambda L_c, \tag{16}$$

where $\lambda$ is a coefficient to balance two losses of network reconstruction and community detection.

At last, we obtain final community divisions from the optimized distribution **Q**. The community label of each node is assigned by:

$$g_i = \arg\max_j(\mathbf{q}_{ij}), \tag{17}$$

where $g_i$ is the community label of node $i$.

### 3.5 Complexity anlaysis

Assume the number of nodes as $N$, the number of layers in a multiplex network as $L$, the depth of GCA as $K$, and the number of detected communities $M$. We consider the computational complexity of each module. The node initial embedding is generated by DeepWalk which contains the training of the Skip-gram model. Given the number of random walks $\rho$, walk length $t$, window size $\omega$, and initial embedding size $d_0$, its computation complexity is in the order of $\mathcal{O}(L\rho Nt\omega(d_0+d_0 logN))$ according to Chen et al. (2018). GCA is a GCN-based model which has linear time complexity with the number of edges. Therefore, we focus on the number of edges in each layer $|E_l|, l = 1, 2, ..., L$ and the encoding dimension of each scale in GCA $d_k, k = 1, 2, ..., K$. The complexity of GCA is in the order of $\mathcal{O}(\sum_{l=1}^{L}|E_l|d_0d_1d_2...d_K)$. The MFN consists of multiple fully connected unit and the initial input is the first scale of GCA. The time complexity of MFN is in the order of $\mathcal{O}(LNd_1^2d_2^2...d_K^2)$. For the self-training module, the computation time mainly depends on Eq.(13). According to Xie et al. (2016), the computational complexity is $\mathcal{O}(NM + NlogN)$. To sum up, the total time complexity of GCFM is in the order of $\mathcal{O}(L\rho Nt\omega(d_0 + d_0 logN) + \sum_{l=1}^{L}|E_l|d_0d_1d_2...d_K + LNd_1^2d_2^2...d_K^2 + NM + NlogN)$, which has a linear relation with the number of layers and edges in each layer.

## 4 Experiment setup

### 4.1 Datasets

We conduct experiments on synthetic datasets and real-world datasets without ground truth community labels, respectively. For synthetic datasets, we use **mLFR** (Bródka 2016) which are controlled by the *mixing parameter* $\mu$ and a smaller $\mu$ suggests more obvious community structures. We will vary it in our experiments. For real-world datasets, we use **AUCs** (Magnani et al. 2013), **RM** (Eagle and Pentland 2006), **C.elegans** (Chen et al. 2006), **London** (De Domenico et al. 2014), **Vickers** (Zhang et al. 2017), **FFTWYT** (Magnani and Rossi 2011), **CKM** (Coleman et al. 1957), **Plasmodium** (Stark et al. 2006), **HumanHIV1** (Stark et al. 2006), and **FriendFeed** (Magnani and Rossi 2011) to conduct experiments. The details of datasets are as follows.

• **mLFR** (Bródka 2016): It is a benchmark tool for generating synthetic multiplex networks. The main parameter of mLFR is the *mixing parameter* $\mu$ which is the probability of a node connecting to another node in a different community. A smaller value of $\mu$ suggests that a multiplex network contains more obvious community structures. Table 1 presents the parameter settings for the mLFR datasets in our experiment.

The following real-world multiplex networks are from different domains, and Table 2 summarizes their statistics.

• **AUCs** (Magnani et al. 2013): It is a 5-layer social network consisting of 61 nodes as employees in the Aarhus University and 620 edges as their social relations, including work together, lunch together, friendship on Facebook, off-line friendship, and co-authorship.

• **RM** (Eagle and Pentland 2006): It is 3-layer social network from the MIT Reality Mining project, which consists of 94 nodes as project participants and 1,385 edges as their social relations: including friendship, average proximity at work and average proximity outside lab.

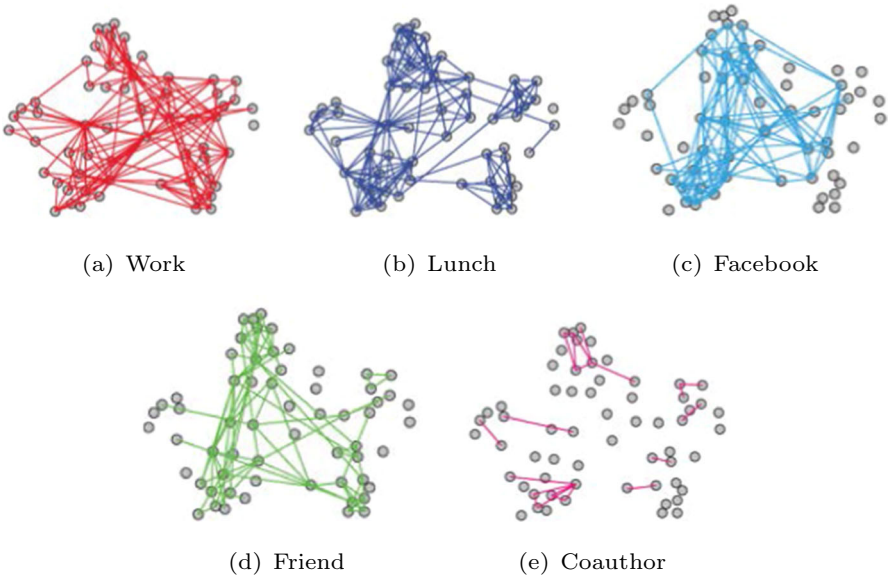
• **C.elegans** (Chen et al. 2006): It is 3-layer neuronal network of the nematode "Caenorhabditis Elegans", consisting of 279 nonpharyngeal neurons and 5,863 synaptic connections, including three types, namely, electric connection, chemical monadic connection and chemical polyadic connection.

• **London** (De Domenico et al. 2014): It is a 3-layer traffic transportation network consisting of 369 nodes and 441 edges, where nodes are stations and three types of edges are transportation lines, including the underground lines, overground lines, and DLR.

• **Vickers** (Zhang et al. 2017): It is a 3-layer offline social network of 29 nodes as seventh grade students in a school and 740 edges for their social relations, including affinity in the class, best friends and working together.

• **FFTWYT** (Magnani and Rossi 2011): It is a 3-layer online social network, consisting of 6,407 users and 74,862 edges. The three layers correspond to three online social platforms, including Friendfeed, Twitter, and YouTube.

• **CKM** (Coleman et al. 1957): It is a 3-layer social network, consisting of 246 nodes and 1,551 edges. The data is collected from physicians in four towns in Illinois,



(a) Work (b) Lunch (c) Facebook

(d) Friend (e) Coauthor

**Fig. 4** The topological structure of each layer in AUCs multiplex network

Peoria, Bloomington, Quincy and Galesburg, which includes the physicians' adoption of a new drug.

• **Plasmodium** (Stark et al. 2006): It is a 3-layer biological network, consisting of 1,203 nodes and 2,521 edges, which includes three types of interactions between genes and proteins, i.e. direct interaction, physical association and association.

• **HumanHIV1** (Stark et al. 2006): It is a 5-layer biological network, consisting of 1,005 nodes and 1,355 edges. It considers different types of genetic interactions about HIV type 4 in the Biological General Repository for Interaction Datasets, i.e. physical association, direct interaction, colocalization, association, and suppressive genetic interaction.

• **FriendFeed** (Magnani and Rossi 2011): It is a 3-layer online social network, consisting of 21,006 nodes and 573,600 edges. It mainly contains interactions among users in Friendfeed collected over the two months, including commenting, liking, and following.

### 4.2 Competitors

We compare our GCFM with the following competitors:

• PMM (Tang et al. 2009) is based on modularity optimization and matrix decomposition, which consists of structural feature extraction and cross-layer integration.

• MDLPA (Boutemine and Bouguessa 2017) introduces a constrained label propagation mechanism for multiplex networks.

• CSNMF (Gligorijević et al. 2019) introduces a collective factorization framework which uses symmetric NMF (Kuang et al. 2012) for each network layer and then

**Table 1** The basic parameter setting of mLFR dataset

| mLFR dataset Parameter | Description | Value |
|---|---|---|
| $N$ | Number of nodes | [500, 2000] |
| $L$ | Number of layers | 4 |
| $d$ | Average degree | 16 |
| $maxd$ | Maximal degree | 32 |
| $\alpha$ | Degree power-law | 2 |
| $\beta$ | Community power-law | 1 |
| $\mu$ | Mixing parameter | [0.2, 0.25, 0.3, 0.35, 0.4] |

**Table 2** The statistics of real datasets

| Datasets | Description | #Nodes | #Edges | #Layers |
|---|---|---|---|---|
| AUCs | Social | 61 | 620 | 5 |
| RM | Social | 94 | 1,385 | 3 |
| C.elegans | Neuronal | 279 | 5,863 | 3 |
| London | Transportation | 369 | 441 | 3 |
| Vickers | Social | 29 | 740 | 3 |
| FFTWYT | Social | 6,407 | 74,862 | 3 |
| CKM | Social | 246 | 1,551 | 3 |
| Plasmodium | Biological | 1,203 | 2,521 | 3 |
| HumanHIV1 | Biological | 1,005 | 1,355 | 5 |
| FriendFeed | Social | 21,006 | 573,600 | 3 |

generates a common feature representation by fusing layers to extract community structures.

• DH-Louvain (Shao et al. 2022) is a multi-greedy algorithm for community detection in multiplex networks which optimizes a weighted modularity density.

• M-DeepWalk (Song and Thiagarajan 2019) first constructs a *supra graph* for a multiplex network and then uses the DeepWalk (Perozzi et al. 2014) to obtain node representations that are input into an auto-encoder to extract cohesive structures in the latent space.

• DMGI (Park et al. 2020) extends Deep Graph Infomax (Velickovic et al. 2019) to multiplex networks and jointly integrate the nodes' embeddings by a consensus regularization framework.

• HDMI (Jing et al. 2021) designs a joint supervision signal which includes both extrinsic and intrinsic mutual information to optimizes node representation in multiplex networks.

### 4.3 Metrics

For synthetic datasets, as they have ground truth labels, we use the commonly used evaluation metrics for supervised learning, including NMI (Normalized Mutual Information), ARI (Adjusted Rand Index), and Purity. For real-word datasets, as they do not contain ground truth labels, we adopt the widely used Modularity metric (Mucha et al. 2010) (also used in synthetic datasets) with two parameters, the resolution parameter $\gamma_s$ and the coupling parameter $\mathscr{C}_{jsr}$. The details of metrics are as follows.

NMI is used to trade-off the quality of communities against the number of communities:

$$NMI(\Omega; C) = \frac{I(\Omega; C)}{[H(\Omega) + H(C)]/2}, \tag{18}$$

where $\Omega$ is the community division and $C$ is the ground truth. $I(\Omega; C)$ is the mutual information between $\Omega$ and $C$. $H(\cdot)$ is the Shannon information entropy.

Rand Index is the percentage of true decisions, defined by

$$RI(\Omega; C) = \frac{TP + TN}{TP + FP + FN + TN}, \tag{19}$$

where true positive ($TP$), true negative ($TN$), false positive ($FP$), and false negative ($FN$) are decisions between $\Omega$ and $C$. ARI is to scale Rand Index in the range of [0,1].

Purity is the percentage of nodes which are classified correctly:

$$Purity(\Omega; C) = \frac{1}{N} \sum_m \max_j |\omega_m \cap c_j|, \tag{20}$$

where $\omega_m$ and $c_j$ mean the $m$-th community in $\Omega$ and the $j$-th community in $C$ respectively.

The multilayer modularity evaluation metric is defined by:

$$Q_m = \frac{1}{2\eta} \sum_{ijsr} [(\mathbf{A}_{ijs} - \gamma_s \frac{d_{is} d_{jr}}{2m_s}) \delta(s, r) + \delta(i, j) \mathscr{C}_{jsr}] \delta\left(g_{is}, g_{jr}\right), \tag{21}$$

where $\mathbf{A}_{ijs}$ is the intra-layer edge strength and $\mathscr{C}_{jsr}$ is the inter-layer edge strength or called the coupling parameter (subscript $i$, $j$ indexes nodes, $s, r$ indexes layers). $\gamma_s$ is a resolution parameter in the $s$-th layer. $\eta = \left(\sum_{ijs} \mathbf{A}_{ijs} + \sum_{jsr} \mathscr{C}_{jsr}\right)/2$ is the total multilayer edge strength. $m_s = \left(\sum_{ij} \mathbf{A}_{ijs}\right)/2$ is the total intra-layer edge strength of the $s$-th layer. $d_{is}$ is the degree of nodes $i$ of the $s$-th layer. $g(\cdot)$ is the community label. $\delta(\cdot)$ is the Kronecker function. In our experiments, the resolution parameter $\gamma_s$ and the coupling parameter $\mathscr{C}_{jsr}$ are set to 1. The multilayer modularity is scaled in the range of [0, 1]. A higher modularity indicates better community detection result.

### 4.4 Parameters

In our GCFM algorithm, we use the DeepWalk to generate an 128-dimensional initial embedding for each node. For the DeepWalk, we set the window size as 10, walk length as 40, and the number of walks started per node as 20 for all datasets. The depth of GCA is set to 4. The dimension of each GCA scale is set to 128-512-1024-2048-10 for synthetic datasets and 128-1024-2048-2048-10 for real-world datasets. The learning rate $r$ is set to 0.001. Considering that the reconstruction loss $L_r$ is too larger than the community detection loss $L_c$, we set the balance coefficient $\lambda = 10$.

For the competitor M-DeepWalk, as it assigns a community label for a node in each layer, we use the majority voting to determine its final community label. The MDLPA can automatically determine the number of communities without a priori assumption and do not need any hyper-parameter. The PMM contains a single hyper-parameter as the number of structural features extracted from each layer. We select the number of structural features in [5, 20] with a step of one per increment and choose the best result. The CSNMF has no external parameters, except a pre-specified community number. We set it the same as the target community number in synthetic datasets and tune it in real-world datasets. The DH-Louvain is also parameter-free and can determine the number of communities by itself. For the M-DeepWalk, we set the threshold of generating a supra graph as 0.1 for the synthetic datasets and 0.2 for the real-world datasets. The parameters of DeepWalk are the same as ours. The learning rate is set to 0.001. For the DMGI and HDMI, we use the initial embedding in Sect. 3.1 as 128-dimensional node features in each layer. The learning rate is also set to 0.001. For all the coefficient of module loss and regularization in the DMGI, we follow the original paper and select from [0.0001, 0.001, 0.01, 0.1] and tune the coefficients to obtain the best result. For the HDMI, we set the layer and fusion coefficients to 1 for all layers in synthetic datasets and use grid search to tune in the real-world datasets. For all the competitors and our model, we run them 10 times and take the average result.

## 5 Results and analysis

### 5.1 Performance on synthetic datasets

Figure 5 compares the community detection performance on mLFR synthetic networks. We observe that our GCFM performs the best in almost all the cases. Furthermore, the performance improvements are much significant compared with the competitors in the cases of $\mu \geq 0.25$. We also notice that the much better modularity of our GCFM over that of competitors suggests stronger modular structures of its detected communities.

These results validate the effectiveness of our GCFM algorithm: The GCA module encodes neighbor-aware structural information for each node to explore locally associated structures at different scales in each network layer. The MFN module fuses structural encodings first for multiple network layers and then for different convolution scales, so as to learn a node representation featuring both intra-layer locally associated structures and inter-layer semantic relations. Finally, the self-training mechanism

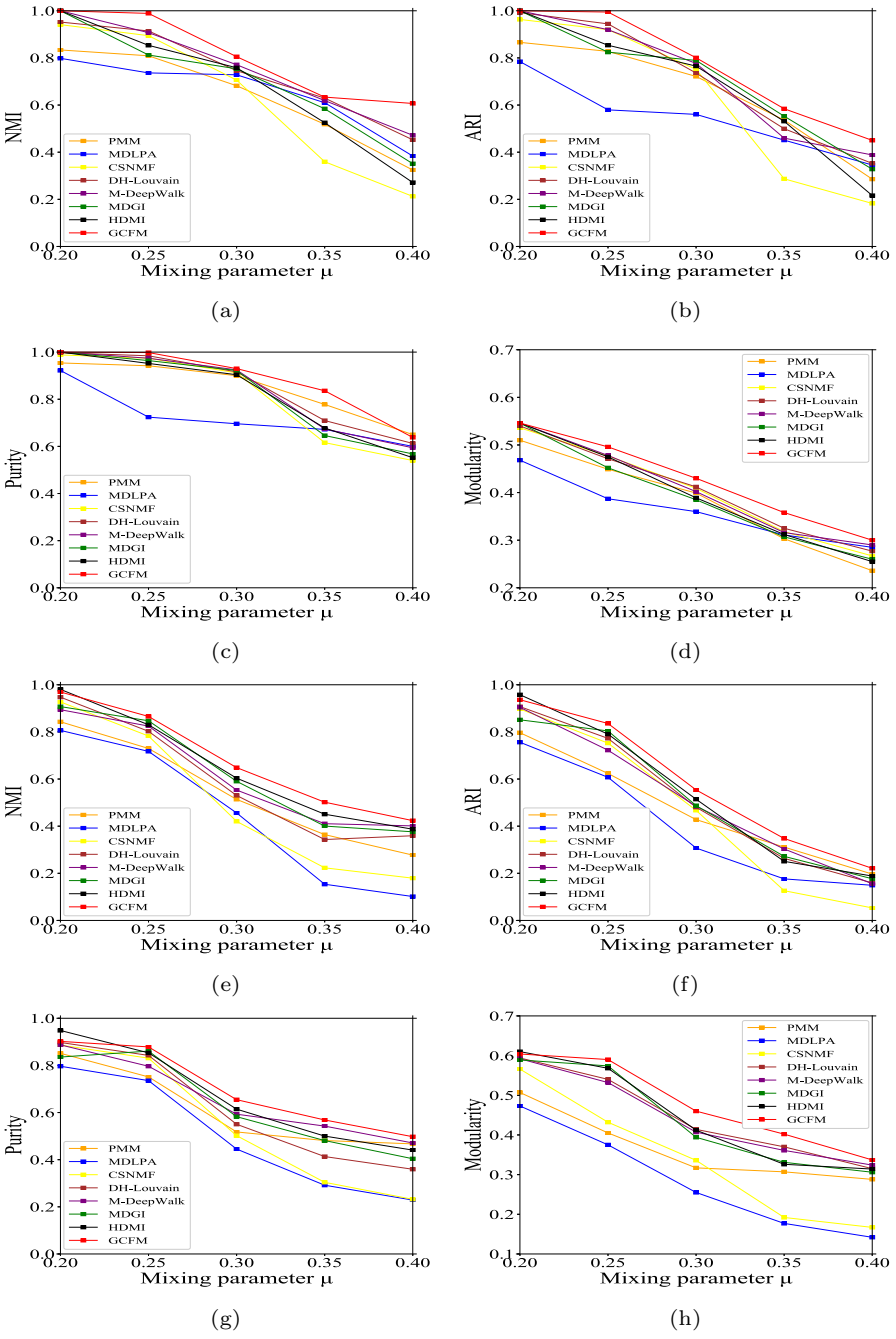



**Fig. 5** The result of NMI, ARI, purity and modularity for mLFR datasets. The upper four subfigures show the varying of **a** NMI, **b** ARI, **c** purity, **d** modularity in small mLFR datasets with 500 nodes and the lower four subfigures are the varying of **e** NMI, **f** ARI, **g** purity, **h** modularity in larger mLFR datasets with 2000 nodes

extracts cohesive communities from the viewpoint of community distribution via the soft assignment, which can iteratively optimize each node representation close to its community center.

Some other analysis for the results are as follows. We can observe that the performance of all algorithms degrade with the increase of the mixing parameter $\mu$. For the mLFR synthetic datasets, a lager value of $\mu$ indicates to include more connections in between two nodes belonging to different communities, which makes it more difficult for the task of community detection. We can also observe that the M-DeepWalk based on node representation learning have a good performance in many cases. The M-DeepWalk extends the DeepWalk (Perozzi et al. 2014) by enabling cross-layer random walks for learning node representations. Even given its simple learning technique, the results indicate that learning node representation could be a powerful approach for the community detection task. For DMGI and HDMI, they are both based on mutual information maximization which can narrow the difference between network layers and have a relatively good performance. The PMM has an average performance by optimizing the principal modularity. The CSNMF, which utilizes collective matrix decomposition, is sensitive to the mixing parameter $\mu$. When $\mu \geq 0.25$, the performance of CSNMF drops sharply. Though MDLPA can determine community number automatically, the losing of multiplex information when flattening multiplex networks leads to the worst performance in most mLFR datasets. The DH-Louvain has two phases, including node merging and community merging, to optimize a weighted modularity density, and its performance is on the average among all competitors.

### 5.2 Performance on real-world datasets

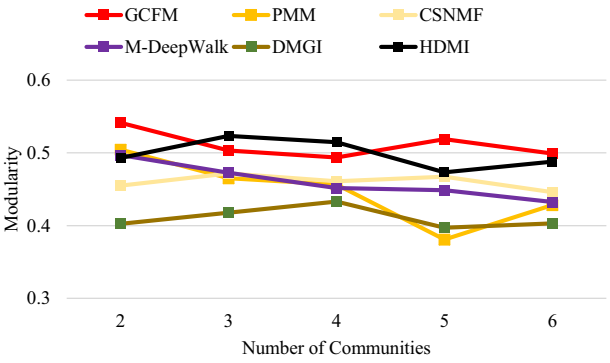
Table 3 presents the modularity results on real-world datasets. The numbers in the brackets are the number of detected communities in these algorithms when achieving their respective best modularity. We note that the PMM, CSNMF, M-DeepWalk, MDGI, HDMI and GCFM algorithms need to pre-specified the number of communities; While MDLPA and DH-Louvain algorithms do not need to do so, and they can automatically determine the number of communities after finishing algorithm execution. Figure 6 presents the modularity results with different pre-specified community numbers in three different types of multiplex networks. We observe that our GCFM outperforms the others for its superiority and robustness.

We again observe that our GCFM achieves the best modularity in most real-world datasets, and the third place in the London dataset. The specialty of the London dataset lies in its sparsity: It is a 3-layer network containing 369 nodes yet with only 441 edges. Indeed, it contains many isolated nodes, each of which cannot link to any other node in one or more network layers. Generally speaking, a community division with high modularity should assign each isolated node an independent community label. MDLPA directly regard isolated nodes as independent communities, which can be found from their much larger community numbers in Table 3. On the other hand, for algorithms needing pre-specified community numbers, they do not discriminate isolated nodes yet still clustering them into communities, which leads to a lower modularity. This phenomenon could be easily addressed by firstly removing isolated

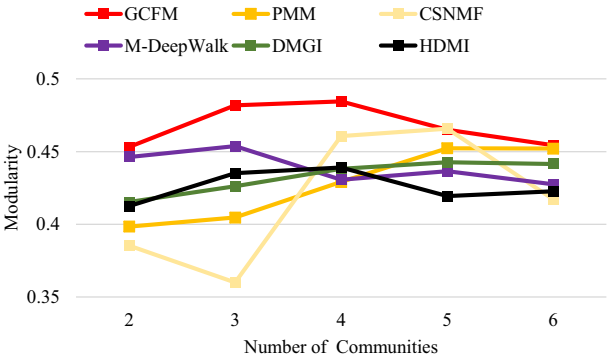
**Table 3** Modularity on real-world datasets

| Dataset | PMM | MDLPA | CSNMF | DH-Louvain | M-DeepWalk | DMGI | HDMI | GCFM |
|---|---|---|---|---|---|---|---|---|
| AUCs | 0.5421 (3) | 0.5512 (6) | 0.6160 (3) | 0.6057 (3) | 0.5049 (3) | 0.5721 (4) | 0.6013 (5) | **0.6309** (3) |
| | ±0.0014 | ±0.0021 | ±0.0025 | ±0.0035 | ±0.0040 | ±0.0017 | ±0.0027 | ±0.0025 |
| RM | 0.5044 (2) | 0.5399 (4) | 0.4718 (3) | 0.5218 (4) | 0.4969 (2) | 0.4331 (4) | 0.5235 (3) | **0.5414** (2) |
| | ±0.0005 | ±0.0013 | ±0.0041 | ±0.0028 | ±0.0022 | ±0.0024 | ±0.0023 | ±0.0011 |
| C.elegans | 0.4523 (5) | 0.4451 (8) | 0.4658 (4) | 0.4403 (4) | 0.4538 (3) | 0.4427 (5) | 0.4391 (4) | **0.4845** (4) |
| | ±0.0034 | ±0.0032 | ±0.0038 | ±0.0026 | ±0.0021 | ±0.0030 | ±0.0023 | ±0.0022 |
| London | 0.7930 (6) | **0.9185** (47) | 0.7915 (5) | 0.8051 (9) | 0.6838 (3) | 0.7042 (4) | 0.7331 (6) | 0.8222 (5) |
| | ±0.0029 | ±0.0027 | ±0.0134 | ±0.0022 | ±0.0016 | ±0.0019 | ±0.0037 | ±0.0018 |
| Vickers | 0.2417 (3) | 0.2255 (2) | 0.2909 (2) | 0.2251 (3) | 0.2102 (3) | 0.2379 (4) | 0.2368 (5) | **0.2923** (3) |
| | ±0.0009 | ±0.0014 | ±0.0021 | ±0.0025 | ±0.0028 | ±0.0014 | ±0.0012 | ±0.0023 |
| FFTWYT | 0.2781 (10) | 0.2849 (65) | 0.2833 (6) | 0.2704 (10) | 0.1866 (3) | 0.2477 (9) | 0.2784(8) | **0.2998** (5) |
| | ±0.0018 | ±0.0026 | ±0.0045 | ±0.0023 | ±0.0034 | ±0.0023 | ±0.0031 | ±0.0026 |
| CKM | 0.4981 (7) | 0.7653 (15) | 0.6515 (5) | 0.6135 (7) | 0.4509 (6) | 0.6408 (4) | **0.7743** (6) | 0.7207 (9) |
| | ±0.173 | ±0.0064 | ±0.0065 | ±0.0052 | ±0.0061 | ±0.0041 | ±0.0033 | ±0.0042 |
| Plasmodium | 0.6395 (9) | 0.6079 (32) | 0.7029 (35) | 0.6511 (14) | 0.5496 (21) | 0.6271 (12) | 0.5888 (14) | **0.7148** (11) |
| | ±0.0032 | ±0.0026 | ±0.0032 | ±0.0028 | ±0.0031 | ±0.0022 | ±0.0024 | ±0.0014 |
| HumanHIV1 | 0.8850 (6) | **0.9597** (20) | 0.9366 (10) | 0.5942 (12) | 0.8264 (4) | 0.8852 (11) | 0.8761(8) | 0.9175 (10) |
| | ±0.0023 | ±0.0027 | ±0.0063 | ±0.0038 | ±0.0040 | ±0.0037 | ±0.0027 | ±0.0023 |
| FriendFeed | 0.2514 (12) | 0.2870 (9) | 0.3166 (12) | 0.3461 (13) | 0.2936 (10) | 0.3147 (8) | 0.3266(5) | **0.3508** (14) |
| | ±0.0009 | ±0.011 | ±0.0013 | ±0.0009 | ±0.0012 | ±0.0017 | ±0.0021 | ±0.0018 |

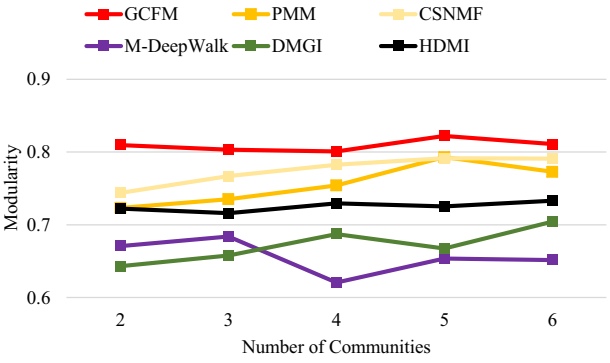
Bold values indicate the best performance among its peers




(a) RM




(b) C.elegans




(c) London

**Fig. 6** Modularity with different pre-specified numbers of communities

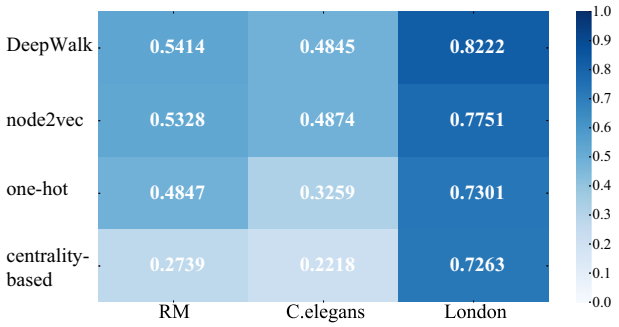



**Fig. 7** Modularity with different ways of generating node initial embedding

nodes for community division. From Fig. 6, it can be observed that our GCFM achieves consistent good performance while some competitors such as the CSNMF and PMM have some performance variations with the increase of the numbers of communities. Both the CSNMF and PMM are based on the adjacency matrix decomposition. Varying community numbers would impact on the eigenvectors during matrix decomposition, which again would impact on the obtained community structures in the process of two medium-sized communities merging into one or a huge community being divided into multiple communities. In our GCFM, the community number only impacts on the self-training module and KL loss.

An interesting observation is that the M-DeepWalk often plays the worst in these real-world datasets, which is contrary to the results in the synthetic datasets. This discrepancy comes from the differences of two types of datasets. In the synthetic datasets, the topological structure of each layer is similar, that is, each node has similar neighborhood in different layers, which makes it easier to find more inter-layer edges. While in real-world datasets, the topological structure of different network layers could differ greatly, which leads to the loss of structural information when generating a supra graph. While our GCFM can decrease such inconsistency of nodes in different layer.

## 5.3 Ablation study

### 5.3.1 Node initial embedding

To investigate the impact of node initial embedding, we compare the DeepWalk with the other three methods, i.e. one-hot, node2vec (Grover and Leskovec 2016), and node centrality. For node2vec, we set its return parameter $p = 1$ and in-out parameter $q = 0.5$. For centrality-based method, we choose four common centrality metrics as the node initial embedding, including degree centrality, betweenness centrality, closeness centrality, and load centrality. Figure 7 presents the modularity results with different node initial embeddings. The two graph embedding algorithms, namely, DeepWalk and node2vec achieve better performance than the other two. This is not unexpected, as they can pre-train node initial embeddings with some topological information.

**Table 4** Modularity with different fusion depths

| | RM | C.elegans | London |
|---|---|---|---|
| GCFM-1 | 0.4265 | 0.4368 | 0.7419 |
| GCFM-2 | 0.4883 | 0.4515 | 0.8032 |
| GCFM-3 | 0.4906 | 0.4710 | 0.8065 |
| GCFM-4 | **0.5414** | **0.4845** | **0.8222** |

Bold values indicate the best performance among its peers

### 5.3.2 Fusion depth analysis

For a given four-scale GCA module, we conduct experiments to analyze the impact of fusion depths. The outputs of the four-scale GCA are denoted by $\mathbf{H}_1$, $\mathbf{H}_2$, $\mathbf{H}_3$, and $\mathbf{H}_4$. Each time we decrease one output scale and input the others into the MFN module. For example, GCFM-3 means that there are three output scales, i.e. $\mathbf{H}_2$, $\mathbf{H}_3$, and $\mathbf{H}_4$, which are input to the MFN module. Besides, GCFM-1 means that only $\mathbf{H}_4$ is used for self-training community detection. Table 4 presents the modularity results with fusion under different scales. We can see that the output of each GCA scale contributes to the final community division and GCFM-4 outperforms the others, which suggests the necessity of multiscale fusion.

### 5.3.3 Model depth analysis

We also conduct experiments to investigate the effect of model depths, i.e., the GCA scales. Figure 8 plots the modularity results against different model depths. We can obverse that the best modularity results are usually obtained with a 4-scale model. When more than four scales are used, the modularity will not keep increasing but even start dropping. Although multiscale fusion is desirable, a larger model depth, i.e., more GCA scales might introduce the so-called over-smoothing problem: After more neighbors are included into the aggregation operation, the neighbor-aware structural encoding start becoming similar across topology-close nodes; While reducing its capability of discriminating local structures.

### 5.3.4 Parameter analysis

In order to investigate the effect of hyper-parameter, i.e. the loss coefficient $\lambda$ and learning rate $r$, we test the modularity with the increase of $\lambda$ from 0 to 100 under different learning rate, i.e. $r \in \{0.0001, 0.001, 0.01, 0.1\}$. Figure 9 shows the modularity results with varying $\lambda$ and $r$ in C.elegans dataset. We can see that the best modularity is obtained under the setting of $\lambda = 10$ and $r = 0.001$. When $\lambda = 0$, our model can not identify high modular structures, which proves the effectiveness of community detection loss $L_c$.

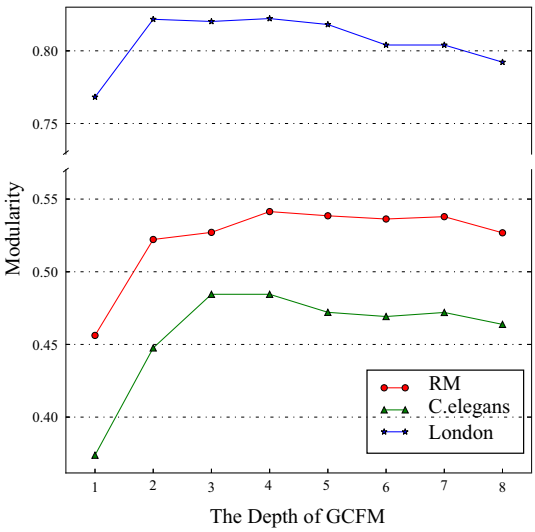



**Fig. 8** Modularity with different model depths

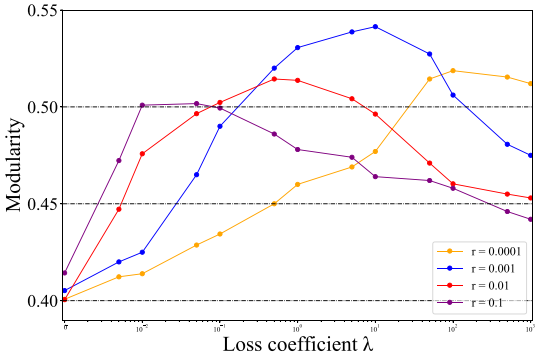



**Fig. 9** Modularity with different $\lambda$ and $r$ in C.elegans dataset

## 5.4 Visualization

### 5.4.1 Network visualization

Figure 10 visualizes an instance of a 3-layer mLFR network with $\mu = 0.35$ and the communities detected by the experimented algorithms. We can observe that the result of our GCFM is closest to the ground truth.

### 5.4.2 Embedding visualization

Figure 11 visualizes the node representations and feature similarity matrix learned by our GCFM for an instance of a 4-layer mLFR network with $\mu = 0.4$. We use the t-SNE (Van der Maaten and Hinton 2008) to visualize the node representation $\mathbf{Z}$ at some training epoches. We can see that the not only the node representations become more evident with the increase of epochs, but also the relations in between nodes, that

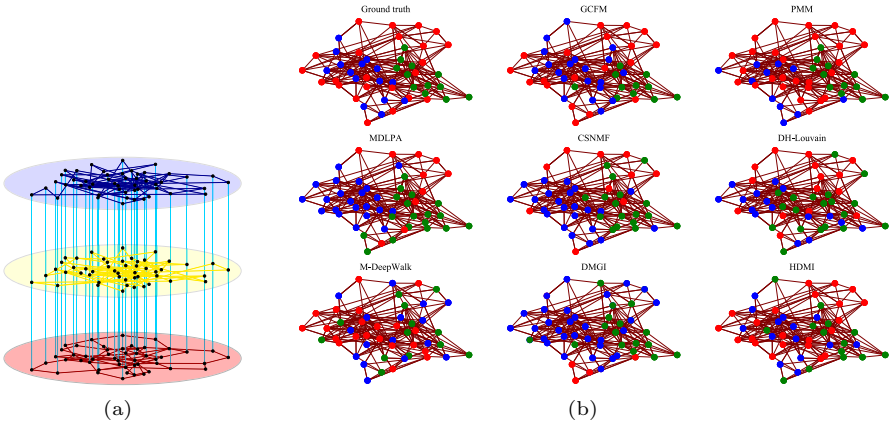



**Fig. 10** Network visualization for an instance of a 3-layer mLFR network with $\mu = 0.35$. **a** The 3-layer mLFR network consists of 50 nodes and 84, 83, 84 edges in the three layers, respectively. The vertical lines illustrate a same node in the three layers. **b** The ground truth of community divisions, as well as the detected communities by the experimented algorithms. Nodes with the same color indicate that they belong to a same community

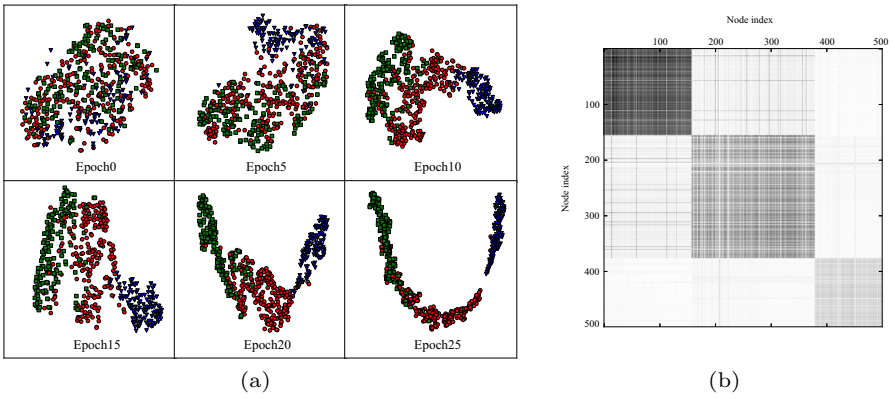



**Fig. 11** Embedding visualization of node representation $\mathbf{Z}$ in a mLFR network with $\mu = 0.4$, consisting of 500 nodes and in total 16,445 edges. **a** 2D visualization of the nodes' representations during the training process of GCFM algorithm. Epoch0 indicates that the representation is only trained with the graph convolutional auto-encoder, and the following epoches have included the self-training component in the representation training. **b** Visualization of feature similarity matrix $\mathbf{ZZ^T}$. For node representation $\mathbf{Z}$ in epoch25, we normalize the matrix $\mathbf{ZZ^T}$ and visualize it with a gray-scale image. The main diagonal blocks indicate the community distribution

is, clearer yet distant community division in the projected space. In the right, the main diagonal blocks of feature similarity matrix $\mathbf{ZZ^T}$ show the community distribution corresponding to the result of the 25-th training epoch.

## 6 Discussion and conclusion

In this paper, we have proposed the GCFM for the task of community detection in multiplex networks. It contains a graph convolutional auto-encoder for encoding neighbor-aware intra-layer structural information, a multiscale fusion network for learning a holistic version of nodes' representations by fusing nodes' encodings at different layers and different scales, and a self-training mechanism to train our model and detect communities. Experiments on both synthetic and real-world datasets have validated the superiority of our GCFM over the state-of-art algorithms. In this work, an excellent feature of our model lies in its full consideration for different scales of local information.

We also obverse some limitations which need to be further investigated. Due to the lack of prior knowledge, we treat each layer equally while each network layer may have its own contribution to the final detected communities in the real world. Therefore, how to judge the weight of each layer in a multiplex network is important. One of the possible approaches is to introduce node attributes. Also the GCFM is mainly used in undirected multiplex networks. Some applications in the real world suggest directed edges. In our future work, we would like to investigate community detection in multiplex networks with node attributes and edge directions.

**Author Contributions** XC: methodology, software, experimentation, result analysis, writing—original draft, visualization. BW: Conceptualization, methodology, result analysis, writing—review editing, supervision.

**Funding** This work is supported in part by National Natural Science Foundation of China (Grant No: 62172167).

**Availability of data and materials** All data generated or analysed during this study are included in this published article.

## Declarations

**Conflict of interest** The authors declare that they have no competing interests.

**Ethics approval** Not applicable.

**Consent to participate** Yes.

**Consent for publication** Yes.

## Appendix A

In this Appendix, we discuss the characteristics of multilayer modularity and the encoding distance with a node corresponding to the same entity in different layers.

**Theorem 1** *Given the modularity of each single layer* $Q_s = \sum_{ij}[(\mathbf{A}_{ijs} - \gamma_s \frac{d_{is}d_{js}}{2m_s})]\delta(g_i, g_j)$*, the multilayer modularity will degenerate into the sum of multiple single layer*

*modularity, i.e.* $\sum_{s} Q_s$*, if a node is assigned with only one community label in its all layers of a multiplex network.*

For the multilayer modularity defined by Eq.(21), it allows that a node in each layer can have its own label; While GCFM and some other baseline methods (PMM, MDLPA, CSNMF, DMGI, HDMI) assign a community label to a node in its belonging layers. In order to analyze the GCFM, we can rewrite Eq.(21) as:

$$Q_m = \frac{1}{2\eta}\sum_{ijsr}[(\mathbf{A}_{ijs} - \gamma_s\frac{d_{is}d_{jr}}{2m_s})\delta(s,r) + \delta(i,j)\mathscr{C}_{jsr}]\delta\left(g_i, g_j\right). \tag{A1}$$

For layer $s, r$ and node $i, j$, the Eq.(A1) can be divided into four cases:

- Case1: $s = r$ and $i = j$ ($\delta\left(g_i, g_j\right) = 1$ and $C_{jsr} = 0$).

$$Q^{(1)} = \frac{1}{2\eta}\sum_{is}[(\mathbf{A}_{iis} - \gamma_s\frac{d_{is}^2}{2m_s})]. \tag{A2}$$

- Case2: $s = r$ and $i \neq j$.

$$Q^{(2)} = \frac{1}{2\eta}\sum_{ijs, i\neq j}\left[\left(\mathbf{A}_{ijs} - \gamma_s\frac{d_{is}d_{js}}{2m_s}\right)\right]\delta\left(g_i, g_j\right). \tag{A3}$$

- Case3: $s \neq r$ and $i = j$ ($\delta\left(g_i, g_j\right) = 1$).

$$Q^{(3)} = \frac{1}{2\eta}\sum_{isr, s\neq r}\mathscr{C}_{jsr}. \tag{A4}$$

- Case4: $s \neq r$ and $i \neq j$.

$$Q^{(4)} = 0. \tag{A5}$$

Therefore, $Q_m$ can be represented as $Q^{(1)} + Q^{(2)} + Q^{(3)} + Q^{(4)}$. $Q^{(3)}$ and $Q^{(4)}$ are constants and $C_{jsr}$ is proven ineffective in our problem setting. It works on the case that each node in each layer is assigned with an independent community label. $Q^{(1)}$ and $Q^{(}(2))$ can be merged as:

$$Q^{(1)} + Q^{(2)} = \frac{1}{2\eta}\sum_{s}\sum_{ij}\left[\left(\mathbf{A}_{ijs} - \gamma_s\frac{d_{is}d_{js}}{2m_s}\right)\right]\delta\left(g_i, g_j\right), \tag{A6}$$

which means the sum of each single layer modularity $Q_s = \sum_{ij}[(\mathbf{A}_{ijs} - \gamma_s\frac{d_{is}d_{js}}{2m_s})]$ $\delta\left(g_i, g_j\right)$. To sum up, our model is to design an optimization function $g(\cdot)$ to maximize

the sum of monoplex modularity:

$$\arg\max_{g} \sum_{s} Q_s. \tag{A7}$$

**Theorem 2** *The GCA can decrease the encoding distance of same node in different layers, i.e.* $\|\hat{\mathbf{h}}_{i,1} - \hat{\mathbf{h}}_{i,2}\|_2$*, if they have similar topological structures in their own layers.*

Given a multiplex network, we focus on the $k$-th GCA scale. Denote $\mathbf{h}_{i,l}$ as the encoding of node $i$ of the $l$-th layer (denoted as $(i,l)$) in $(k-1)$-th scale and we have $\hat{\mathbf{h}}_{i,l} = ReLU(\sum_{j\in N_{i,l}} \frac{\mathbf{h}_{j,l}}{\sqrt{d_{i,l}}\sqrt{d_{j,l}}}\mathbf{W})$, where $N_{i,l}$ is the neighborhoods of node $(i,l)$. Assume $ReLU(\cdot)$ as $\sigma(x) = x$ and $\mathbf{W} = \mathbf{I}$. $\hat{\mathbf{h}}_{i,l}$ can be regarded as the average of neighbor encodings. Now we focus on node $i$ in a two-layer multiplex network, i.e. $\hat{\mathbf{h}}_{i,1}$ and $\hat{\mathbf{h}}_{i,2}$. For $\hat{\mathbf{h}}_{i,1}$, it can be divided into three parts: (a) Self node encoding $\frac{\mathbf{h}_{i,l}}{d_{i,l}}$. (b) The encodings of generalized common neighbors of $(i,1)$ and $(i,2)$, where generalized common neighbors mean that nodes correspond to the same entity. It can be represented as $\frac{\mathbf{B}_1}{\sqrt{d_{i,1}}}$, where $\mathbf{B}_1 = \sum_{u\in N_{i,1}\cap N_{i,2}} \frac{\mathbf{h}_{u,1}}{\sqrt{d_{u,1}}}$ is the sum of common neighbor encodings. (c) The other non-common neighbor encodings, denoted as $\frac{\mathbf{D}_1}{\sqrt{d_{i,1}}}$, where $\mathbf{D}_1 = \sum_{v\in N_{i,1}-N_{i,1}\cap N_{i,2}} \frac{\mathbf{h}_{v,1}}{\sqrt{d_{v,1}}}$. Assume each node in the two layer has the same initial encoding in $(k-1)$-th scale, i.e. $\mathbf{h}_{i,1} = \mathbf{h}_{i,2}$. Then we measure the distance of $\hat{\mathbf{h}}_{i,1}$ and $\hat{\mathbf{h}}_{i,2}$:

$$\begin{aligned}
&\|\hat{\mathbf{h}}_{i,1} - \hat{\mathbf{h}}_{i,2}\|_2 \\
&= \|\left(\frac{\mathbf{h}_{i,1}}{d_{i,1}} - \frac{\mathbf{h}_{i,2}}{d_{i,2}}\right) + \left(\frac{\mathbf{B}_1}{\sqrt{d_{i,1}}} - \frac{\mathbf{B}_2}{\sqrt{d_{i,2}}}\right) + \left(\frac{\mathbf{D}_1}{\sqrt{d_{i,1}}} - \frac{\mathbf{D}_2}{\sqrt{d_{i,2}}}\right)\|_2 \\
&\le \left\|\frac{\mathbf{h}_{i,1}}{d_{i,1}} - \frac{\mathbf{h}_{i,2}}{d_{i,2}}\right\|_2 + \left\|\frac{\mathbf{B}_1}{\sqrt{d_{i,1}}} - \frac{\mathbf{B}_2}{\sqrt{d_{i,2}}}\right\|_2 + \|\frac{\mathbf{D}_1}{\sqrt{d_{i,1}}} - \frac{\mathbf{D}_2}{\sqrt{d_{i,2}}}\|_2 \\
&\le \left|\frac{d_{i,2} - d_{i,1}}{d_{i,1}d_{i,2}}\right| \|\mathbf{h}_{i,1}\|_2 \\
&\quad + \sum_{u\in N_{i,1}\cap N_{i,2}} \left|\frac{\sqrt{d_{i,2}d_{u,2}} - \sqrt{d_{i,1}d_{u,1}}}{\sqrt{d_{i,1}d_{u,1}d_{i,2}d_{u,2}}}\right| \|\mathbf{h}_{u,1}\|_2 \\
&\quad + \left(\left\|\frac{\mathbf{D}_1}{\sqrt{d_{i,1}}}\right\|_2 + \left\|\frac{\mathbf{D}_2}{\sqrt{d_{i,2}}}\right\|_2\right).
\end{aligned} \tag{A8}$$

From Eq.(A8) we can derive the upper bound of the distance between $\hat{\mathbf{h}}_{i,1}$ and $\hat{\mathbf{h}}_{i,2}$. The first term measures the degree difference of $(i,1)$ and $(i,2)$, where $|\frac{d_{i,2}-d_{i,1}}{d_{i,1}d_{i,2}}| \le 1$. If $d_{i,1} \approx d_{i,2}$, the effect of the first term can be ignored; While if the two nodes have no similar degrees such as $d_{i,2} \gg d_{i,1} = 1$, it may achieve the upper bound 1.

The second term represents the influence of the common neighbors. For each item $|\frac{\sqrt{d_{i,2}d_{u,2}}-\sqrt{d_{i,1}d_{u,1}}}{\sqrt{d_{i,1}d_{u,1}d_{i,2}d_{u,2}}}| \leq 1$, we assume $(i,1)$ and $(i,2)$ have similar degrees. Specially, if $d_{u,2} \gg d_{u,1}$, this item will become $|\frac{1}{\sqrt{d_{i,1}d_{u,1}}}|$ which is controlled by the degree of node $(i,1)$ and the degree of its common neighbor $(u,1)$, both normally larger than 1. So usually $|\frac{\sqrt{d_{i,2}d_{u,2}}-\sqrt{d_{i,1}d_{u,1}}}{\sqrt{d_{i,1}d_{u,1}d_{i,2}d_{u,2}}}| < 1$. The third term shows the influence of the non-common neighbors. For the upper bound of each item $|\frac{1}{\sqrt{d_{i,1}d_{v,1}}}|$ and $|\frac{1}{\sqrt{d_{i,2}d_{v,1}}}|$, if $d_{v,1} \gg d_{i,1}$ and $d_{v,2} \gg d_{i,2}$, the effect of the third term can be ignored. Besides, there is certain restrictive correlation between the second term and the third term, which is $d_{i,1} = |u \in N_{i,1} \cap N_{i,2}| + |v \in N_{i,1} - N_{i,1} \cap N_{i,2}|$.

In a multiplex network, we hope that the same node in different layers have approximate performance if they have similar topology so that we can fuse their encodings and obtain a consistent community label. Equation(A8) proves that our GCA can decrease the encoding distance of same node in different layers with similar topology, i.e node degree, common neighbor and one-hop neighbor degree, which will be benefit for the task of community detection.

## References


Ali HT, Liu S, Yilmaz Y, Couillet R, Rajapakse I, Hero A (2019) Latent heterogeneous multilayer community detection. In: Proceedings of the international conference on acoustics, speech and signal processing, pp 8142–8146. IEEE

Berlingerio M, Coscia M, Giannotti F (2011) Finding and characterizing communities in multidimensional networks. In: Proceedings of international conference on advances in social networks analysis and mining, pp 490–494. IEEE

Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech: Theory Exp 2008(10):10008

Bouguessa M, Wang S, Dumoulin B (2010) Discovering knowledge-sharing communities in question-answering forums. ACM Trans Knowl Discov Data 5(1):1–49

Boutemine O, Bouguessa M (2017) Mining community structures in multidimensional networks. ACM Trans Knowl Discov Data 11(4):1–36

Bo D, Wang X, Shi C, Zhu M, Lu E, Cui P (2020) Structural deep clustering network. In: Proceedings of the World Wide Web conference, pp 1400–1410

Bródka P (2016) A method for group extraction and analysis in multilayer social networks. arXiv preprint arXiv:1612.02377

Cao J, Jin D, Yang L, Dang J (2018) Incorporating network structure with node contents for community detection on large networks using deep learning. Neurocomputing 297:71–81

Chang H, Feng Z, Ren Z (2016) Community detection using dual-representation chemical reaction optimization. IEEE Transn Cybern 47(12):4328–4341

Chen BL, Hall DH, Chklovskii DB (2006) Wiring optimization can relate neuronal structure and function. Proc Natl Acad Sci 103(12):4723–4728

Chen Z, Chen C, Zheng Z, Zhu Y (2019) Tensor decomposition for multilayer networks clustering. In: Proceedings of the AAAI conference on artificial intelligence, pp 3371–3378

Chen H, Perozzi B, Hu Y, Skiena S (2018) Harp: Hierarchical representation learning for networks. In: Proceedings of the AAAI conference on artificial intelligence, vol 32

Coleman J, Katz E, Menzel H (1957) The diffusion of an innovation among physicians. Sociometry 20(4):253–270

De Domenico M, Solé-Ribalta A, Gómez S, Arenas A (2014) Navigability of interconnected networks under random failures. Proc Natl Acad Sci 111(23):8351–8356

Eagle N, Pentland AS (2006) Reality mining: sensing complex social systems. Pers Ubiquit Comput 10(4):255–268
Fortunato S (2010) Community detection in graphs. Phys Rep 486(3–5):75–174
Gao X, Zheng Q, Verri FA, Rodrigues RD, Zhao L (2019) Particle competition for multilayer network community detection. In: Proceedings of the 11th international conference on machine learning and computing, pp 75–80
Garcia JO, Ashourvan A, Muldoon S, Vettel JM, Bassett DS (2018) Applications of community detection techniques to brain graphs: algorithmic considerations and implications for neural function. Proc IEEE 106(5):846–867
Gligorijević V, Panagakis Y, Zafeiriou S (2019) Non-negative matrix factorizations for multiplex network analysis. IEEE Trans Pattern Anal Mach Intell 41(4):928–940
Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 855–864
Huang X, Chen D, Ren T, Wang D (2021) A survey of community detection methods in multilayer networks. Data Min Knowl Disc 35(1):1–45
Interdonato R, Tagarelli A, Ienco D, Sallaberry A, Poncelet P (2017) Local community detection in multilayer networks. Data Min Knowl Disc 31(5):1444–1479
Jia Y, Zhang Q, Zhang W, Wang X (2019) Communitygan: Community detection with generative adversarial nets. In: Proceedings of the World Wide Web conference, pp 784–794
Jing B, Park C, Tong H (2021) Hdmi: High-order deep multiplex infomax. In: Proceedings of the web conference 2021, pp 2414–2424
Jin D, Liu Z, Li W, He D, Zhang W (2019) Graph convolutional networks meet markov random fields: Semi-supervised community detection in attribute networks. In: Proceedings of the AAAI conference on artificial intelligence, pp 152–159
Kuang D, Ding C, Park H (2012) Symmetric nonnegative matrix factorization for graph clustering. In: Proceedings of the international conference on data mining, pp 106–117. SIAM
Liu Q, Wang B (2022) Neural extraction of multiscale essential structure for network dismantling. Neural Netw 154:99–108
Liu Q, Wang B, Qi J, Deng X (2022) A new centrality measure based on neighbor loop structure for network dismantling. Digit Commun Netw
Liu F, Xue S, Wu J, Zhou C, Hu W, Paris C, Nepal S, Yang J, Yu PS (2020) Deep learning for community detection: Progress, challenges and opportunities. In: Proceedings of the 29th international joint conference on artificial intelligence, pp. 4981–4987. International Joint Conferences on Artificial Intelligence Organization
Ma X, Dong D, Wang Q (2018) Community detection in multi-layer networks using joint nonnegative matrix factorization. IEEE Trans Knowl Data Eng 31(2):273–286
Magalingam P, Davis S, Rao A (2015) Using shortest path to discover criminal community. Digit Investig 15:1–17
Magnani M, Hanteer O, Interdonato R, Rossi L, Tagarelli A (2021) Community detection in multiplex networks. ACM Comput Surv 54(3):1–35
Magnani M, Micenkova B, Rossi L (2013) Combinatorial analysis of multiple networks. arXiv preprint arXiv:1303.4986
Magnani M, Rossi L (2011) The ml-model for multi-layer social networks. In: ASONAM, pp 5–12. IEEE Computer Society
Mercorio F, Mezzanzanica M, Moscato V, Picariello A, Sperli G (2019) Dico: A graph-db framework for community detection on big scholarly data. IEEE Trans Emerg Top Comput 9(4):1987–2003
Mucha PJ, Richardson T, Macon K, Porter MA, Onnela J-P (2010) Community structure in time-dependent, multiscale, and multiplex networks. Science 328(5980):876–878
Newman ME, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69(2):026113
Park C, Kim D, Han J, Yu H (2020) Unsupervised attributed multiplex network embedding. Proc AAAI Conf Artif Intell 34:5371–5378
Paul S, Chen Y (2022) Null models and community detection in multi-layer networks. Sankhya A 84(1):163–217
Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 701–710

Pramanik S, Tackx R, Navelkar A, Guillaume J-L, Mitra B (2017) Discovering community structure in multilayer networks. In: 2017 IEEE international conference on data science and advanced analytics (DSAA), pp 611–620. IEEE

Raghavan UN, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E 76(3):036106

Shao Z, Ma L, Lin Q, Li J, Gong M, Nandi AK (2022) Pmcdm: privacy-preserving multiresolution community detection in multiplex networks. Knowl-Based Syst 244:108542

Song H, Thiagarajan JJ (2019) Improved deep embeddings for inferencing with multi-layered graphs. In: Proceedings of the international conference on big data, pp 5394–5400. IEEE

Souravlas S, Anastasiadou S, Katsavounis S (2021) A survey on the recent advances of deep community detection. Appl Sci 11(16):7179

Sperlí G (2019) A deep learning based community detection approach. In: Proceedings of the 34th ACM/SIGAPP symposium on applied computing, pp 1107–1110

Stark C, Breitkreutz B-J, Reguly T, Boucher L, Breitkreutz A, Tyers M (2006) Biogrid: a general repository for interaction datasets. Nucleic Acids Res 34(suppl_1), 535–539

Suthers D, Fusco J, Schank P, Chu K-H, Schlager M (2013) Discovery of community structures in a heterogeneous professional online network. In: 2013 46th Hawaii international conference on system sciences, pp 3262–3271. IEEE

Su X, Xue S, Liu F, Wu J, Yang J, Zhou C, Hu W, Paris C, Nepal S, Jin D, et al (2022) A comprehensive survey on community detection with deep learning. IEEE Trans Neural Netw Learn Syst

Tagarelli A, Amelio A, Gullo F (2017) Ensemble-based community detection in multilayer networks. Data Min Knowl Disc 31(5):1506–1543

Tang L, Wang X, Liu H (2009) Uncoverning groups via heterogeneous interaction analysis. In: Proceedings of the 9th international conference on data mining, pp 503–512. IEEE

Tu C, Zeng X, Wang H, Zhang Z, Liu Z, Sun M, Zhang B, Lin L (2018) A unified framework for community detection and network representation learning. IEEE Trans Knowl Data Eng 31(6):1051–1065

Van der Maaten L, Hinton G (2008) Visualizing data using t-sne. J Mach Learn Res 9(11):2579–2605

Velickovic P, Fedus W, Hamilton WL, Liò P, Bengio Y, Hjelm RD (2019) Deep graph infomax. ICLR (Poster) 2(3):4

Wang C, Pan S, Hu R, Long G, Jiang J, Zhang C (2019) Attributed graph clustering: a deep attentional embedding approach. In: Proceedings of the 28th international joint conference on artificial intelligence, pp 3670–367. International Joint Conferences on Artificial Intelligence Organization

Xia L, Huang C, Xu Y, Dai P, Zhang X, Yang H, Pei J, Bo L (2021) Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation. Proc AAAI Conf Artif Intell 35:4486–4493

Xie Y, Gong M, Wang S, Yu B (2018) Community discovery in networks with deep sparse filtering. Pattern Recogn 81:50–59

Xie J, Girshick R, Farhadi A (2016) Unsupervised deep embedding for clustering analysis. In: Proceedings of the international conference on machine learning, pp 478–487. PMLR

Ye F, Chen C, Zheng Z (2018) Deep autoencoder-like nonnegative matrix factorization for community detection. In: Proceedings of the 27th ACM international conference on information and knowledge management, pp 1393–1402

Zhang H, Wang C-D, Lai J-H, Philip SY (2017) Modularity in complex multilayer networks with multiple aspects: a static perspective. In: Proceedings of the applied informatics, pp 1–29. SpringerOpen