Feature Graph Augmented Network Representation for Community Detection

Lei Zhang, Zeqi Wu, Haipeng Yang, Wuji Zhang, and Peng Zhou, *Senior Member, IEEE*

***Abstract*—Community detection plays an important role in understanding complex networks. Many traditional embedding-based community detection methods only focus on the relations between nodes in the topology space (i.e., topology graph). Besides, there are also some works that consider the feature embedding of nodes to further improve the detection performance. However, most of them ignore the relationships between nodes in the feature space (i.e., feature graph). To address this issue, in this article, we construct the feature graph from the features of nodes to capture the relations between nodes in the feature space, and incorporate it with the topology graph and the feature embedding, leading to the novel feature graph augmented network representation for community detection (FGCD) method. Specifically, FGCD extracts the embeddings of topology graph, node features, and feature graph, respectively, and ensembles them by a layerwise fusion method with an attention mechanism. Extensive experiments on 11 real-world datasets show that FGCD outperforms most existing state-of-the-art algorithms, which well demonstrates its superiority.**

***Index Terms*—Attributed networks, community detection, feature graph, network representation learning.**

I. Introduction

COMPLEX systems in reality can often be represented as networks, and networks can often be seen as composed of different communities. Mining the structure of communities is important to understand complex systems, and thus community detection has been widely studied for many years [1], [2], [3].

Community detection, aiming to categorize nodes into various clusters, is a fundamental problem in graph analysis. It has been applied to many tasks such as biochemical [4] and social network analysis [5]. Community detection usually considers two aspects of information: the relations between nodes in the topology space (i.e., topology graph) and features of nodes. For example, in an academic network, authors represent nodes in the network and coauthors' collaborative relationships represent edges in the network. This is the topological structure. Besides, each node also has its features, such as the author ID and the author's research topic. Authors with the same research topic may be associated with a community, and community detection aims to find this community structure according to the topology graph and node features.

The main focus of most early works [6], [7], [8], [9], [10] on community detection is the topology of graphs. For example, Pan et al. [11] applied deep learning to community detection for the first time. To learn graph embeddings, Cavallari et al. [12] provided a novel joint modeling approach that makes use of information loops between node embedding, community detection, and community embedding. Ye et al. [13] introduced an approach that integrates AE modules and NMF modules for complex network analysis. Besides, some works [14], [15], [16], [17], [18] learn the feature embedding and incorporate node features into the community detection task. For example, Li et al. [19] presented an embedding-based model that proposes a community structure embedding method to encode community structures. In recent years, graph convolutional network (GCN) [20] is increasingly focused on the network representation field, and there are many works [21], [22], [23] that improve on GCN. He et al. [24] presented a new method for unsupervised community detection in GCN-based attribute networks. Recently, Qiu et al. [25] proposed a modularity and network structure-based joint optimization method.

However, although they have achieved good performance on community detection tasks, all the above works ignore the relations between nodes in the feature space (i.e., feature graph). On one hand, the feature graph complements the topology graph because the topology graph does not consider any information in the feature space; on the other hand, feature embedding often handles nodes in the feature space independently and also ignores the relations between nodes in the feature space. In fact, feature graphs, topology graphs, and attribute information are three different concepts. As an example, Fig. 1(a) represents the real network partition that nodes 1, 2, 3, and 6 belong to a community, nodes 4 and 5 belong to a community, and nodes 7, 8, and 9 belong to a community. Fig. 1(b) represents the obtained network partition based on the topology graph, which

Manuscript received 27 November 2023; revised 28 March 2024; accepted 25 April 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 61976001 and Grant 62176001, and in part by the Key Projects of University Excellent Talents Support Plan of Anhui Provincial Department of Education under Grant gxyqZD2021089. *(Corresponding author: Peng Zhou.)*

The authors are with the Information Materials and Intelligent Sensing Laboratory of Anhui Province, Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Hefei 230039, China (e-mail: zl@ahu.edu.cn; a2467449838@163.com; haipengyang@126.com; e21201077@stu.ahu.edu.cn; zhoupeng@ahu.edu.cn).

Digital Object Identifier 10.1109/TCSS.2024.3399210

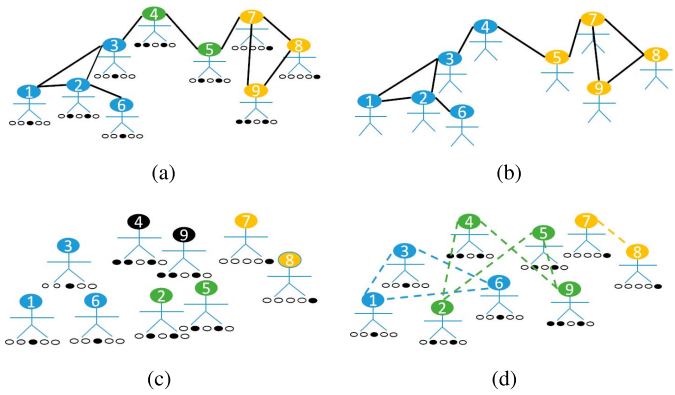

Fig. 1. (a) Division of the original network. (b) Topology graph-based network partitioning. (c) Attribute information-based network partitioning. (d) Feature graph-based network partitioning. It can be observed that in (b), nodes 4 and 5 are not correctly partitioned into the correct communities. However, in (d), they are correctly partitioned. Therefore, it can be inferred that the feature graph can explore network structures that are not considered by the topology graph.

incorrectly partitions nodes 4 and 5 into other communities. Fig. 1(c) represents the obtained network partition based on the attribute information, which incorrectly partitions nodes 2, 4, and 9 into other communities. Fig. 1(d) represents the obtained network partition based on the feature graph constructed by using the node's attribute information, which correctly partitions nodes 4 and 5 into the correct communities and well compensates for the incorrect partition of nodes 4 and 5 in topology graph. It is clear that using the constructed feature graph could augment the traditional community detection by only using topology graph and attribute information. Therefore, how to properly combine these three aspects of information is particularly important to the community detection problem.

To tackle this challenge, this article introduces a novel feature graph augmented network representation for community detection (FGCD) method, which fully considers and fuses all three aspects of topology graph, feature embedding, and feature graph for community detection. Specifically, first, the application of GCN lies in learning representations of topology graph. This is because GCN can aggregate information from neighboring nodes through their convolutional operations, thereby capturing the structural characteristics of the network. Second, autoencoder (AE) is utilized for learning representations of node features. Despite the topology graph providing connectivity information between nodes, the topology graph overlooks the features inherent to each node. AE can learn compact representations of data in an unsupervised manner, thereby capturing latent patterns within node features, which serves as a complement to topology graph information. Last, the introduction of feature graphs aims to further capture relationships between nodes in feature space. By constructing a k-nearest neighbor graph as a feature graph and utilizing graph autoencoder (GAE) to learn the node representations, information from neighboring nodes in feature space can be aggregated. Only by considering the topology graph or feature embeddings, it is hard to understand the similarity and relationships between nodes in the feature space. To fuse these three aspects more profoundly, a layerwise fusion strategy is carefully designed by exploiting the attention mechanism to obtain the final representation. At last, a self-supervised strategy is provided to train the network to obtain a reliable community detection result.

The key contributions of the method are emphasized as follows.

1) We leverage node relationships in the feature space to enhance community detection. By capturing these relationships, we accurately identify the community structure within the network, improving the precision of community detection.
2) We design a novel framework that seamlessly integrates topology graphs, feature embeddings, and feature graph representations in a hierarchical manner. To the best of our knowledge, this is the first work to innovatively combine these three elements for community detection.
3) We conduct extensive experiments on 11 real datasets to demonstrate the effectiveness of FGCD compared with the state-of-the-art algorithms.

The rest of the article is organized as follows. In Section II, the related work of community detection is introduced. In Section III, the concepts related to the problem are provided. In Section IV, the construction of the FGCD model is presented. The comparison of FGCD with ten state-of-the-art algorithms on 11 real datasets is given in Section V. The conclusions and future perspectives of this article are provided in Section VI.

II. Related Work

Community detection methods can be roughly categorized into two classes: the topological structure-based community detection and the community detection incorporating node features, as shown in Table I.

The topological structure-based methods focus on the relations between nodes in the topology space. They often use the network embedding for community detection. For example, Perozzi et al. [26] learned the hidden information of the network and represent the nodes in the graph as a vector containing the underlying information. Tian et al. [27] proposed a novel method that initially employs stacked AEs to acquire a nonlinear network embedding of the topology graph. Subsequently, the k-means algorithm is applied to the network embedding to obtain the community detection results. Tang et al. [28] suggested a community detection method that can handle both directed and undirected graphs with or without weights. Wang et al. [29] proposed a structural deep network embedding method that utilizes a multilayer nonlinear function that also preserves the network structure by utilizing low-order proximity, thus enabling the capture of highly nonlinear network structures. Grover and Leskovec [30] designed a model that maintains node neighbor information. Cao et al. [31] proposed a new model for learning graph embedding, by employing a random surfing model to capture the topology of the graph directly, instead of employing a sampling-based approach to

TABLE I
CATEGORIES OF COMMUNITY DETECTION ALGORITHMS

Method	Topology Graph	Node Feature	Feature Graph
Tang et al. [28]	√		
Wang et al. [29]	√		
Grover et al. [30]	√		
Cao et al. [31]	√		
Wang et al. [32]	√		
Zhu et al. [33]	√		
Bo et al. [34]	√	√	
Berahmand et al. [35]	√	√	
Mrabah et al. [36]	√	√	
Lu et al. [37]	√	√	
Gong et al. [38]	√	√	
Fanseu et al. [39]	√	√	
Shang et al. [40]	√	√	
Berahmand2 et al. [41]	√	√	
Berahmand et al. [42]	√	√	
FGCD (our)	√	√	√

obtain the topology of the graph. Wang et al. [32] proposed a matrix factorization-based model modularized nonnegative matrix factorization, which learns the network structure features by nonnegative matrix factorization and uses the matrix expression formula of modularity to constrain the community structure. Recently, Zhu et al. [33] proposed a community detection network embedding method based on NMF, which integrates node and structure similarities into a unified similarity matrix in a low-dimensional space to improve the quality of community detection. However, all these methods ignore the node features. In fact, node features can provide more useful information that is not contained in the topological structure.

The methods that incorporate node features consider both the topology and node features. For example, Bo et al. [34] combined the AE and GCN with a novel transfer operator and a double self-supervised module. Shchur and Günnemann [35] utilized graph convolutional neural network to extract network embeddings and then exploit the Bernoulli–Poisson (BP) probabilistic model to mine the community structure. Berahmand et al. [36] proposed a method based on augmented graph regularization NMF that combines augmented attribute graphs with topological structure and attribute information. Mrabah et al. [37] designed a new framework that mitigates feature drift and feature randomness effects. Lu et al. [38] introduced a new generative model and designs a colearning method that assembles the nested EM algorithm and belief propagation to train both parts of the model. Gong et al. [39] proposed a dual redundancy reduction-based attribute graph clustering algorithm to reduce the information redundancy in the latent space and introduced an adversarial learning mechanism to ensure the diversity of the compared sample pairs. Fanseu et al. [40] proposed an extended graph convolution framework for the attribute graph clustering task, providing a general and interpretable solution for clustering over many different types of attribute graphs, including undirected, directed, heterogeneous, and hyperattribute graphs. Shang et al. [41] proposed an attribute community detection algorithm that combines graph regularization, nonnegative matrix factorization, and representation learning. The algorithm primarily leverages both topological and attribute information. To address the issues of dimensionality reduction via linear methods and random selection of side information in semisupervised community detection methods, Berahmand et al. [42] leveraged a semi-AE with a defined pairwise constraint matrix based on pointwise mutual information in the representation layer to accurately learn distinctive features. Recently, Berahmand et al. [43] effectively addressed the challenges of ignoring attribute information, neglecting geometric data points structures, and failing to distinguish irrelevant features and data outliers in attributed graph clustering by incorporating node attribute similarity, graph regularization, and sparsity constraints. More relevant research on graph convolutional neural networks and graph AEs can be found in recent surveys by Jin et al. [44] and Su et al. [45].

However, although the above works have achieved promising results, the relationship between nodes in the feature space is ignored. To tackle this problem, in this article, a new FGCD method is proposed by fully considering the relations in the feature space. Moreover, a layerwise fusion strategy is designed to better incorporate the relations in the feature space with the topology graph and the node features.

III. PROBLEM FORMULATION AND NOTATIONS

The proposed method learns the embeddings of nodes from network representations and then utilizes the embeddings to address downstream task community detection. This section introduces the core concepts relevant to the problem addressed by the method FGCD and defines the notation used throughout the article.

A. Community Detection

The number of communities is regarded as a priori knowledge, denoted as C. Community detection means that nodes that are connected to each other in the topology space and have similar features are grouped into a community denoted as $C_{ij} = 1$ (i.e., node i belongs to the jth community), and otherwise, for different community denoted as $C_{ij} = 0$. The community affiliation Z_{ij} of each node is derived from the

TABLE II
NOTATIONS

Notations	Meaning
G	The undirected network
V	The set of nodes
E	The set of edges
$\mathbf{A}^t$	The adjacency matrix of the topology graph
$\mathbf{A}^f$	The adjacency matrix of the feature graph
A_{ij}^t	The ith node is connected to the jth node of the topology graph
A_{ij}^f	The ith node is connected to the jth node of the feature graph
$\hat{\mathbf{A}}^f$	The reconstructed adjacency matrix of the feature graph
$\mathbf{X}$	The node features
$\hat{\mathbf{X}}$	The reconstructed node features
C	The number of communities
$\mathbf{H}_L$	The representation in the lth layer of the AE
$\mathbf{Z}_L^t$	The representation in the lth layer of the GCN
$\mathbf{Z}_L^f$	The representation in the lth layer of the GAE
$\mathbf{Z}_{ij}$	The probability of the node i belongs to the community j
$\mathbf{X}_{ij}$	The jth feature of the ith node
$\widetilde{\mathbf{A}}_t$	The self-connected adjacency matrix of topology graph
$\widetilde{\mathbf{A}}_f$	The self-connected adjacency matrix of feature graph

network representation. The final division of communities is obtained by utilizing the community detection algorithm for the embedding of nodes.

B. Problem Settings

In this article, we concentrate on community detection on the attributed graph G, and let $G = \{V, E, \mathbf{X}\}$ denotes an undirected network, in which $V = \{v_1, \ldots, v_n\}$ represents the set of nodes, $E = \{e_{ij}\}$ represents the set of edges, and $i, j \in \{1, \ldots, n\}$. Besides, $\mathbf{X}$ represents the set of features of nodes where X_{ij} denotes the jth feature of the ith node. The goal of network representation is to produce a continuous, low-dimensional representation $\mathbf{z}_j \in \mathbb{R}^d$ for each node, where the representation's dimension is $d\,(d \leq n)$. Besides, $\mathbf{A}^t$ denotes the node relationships of the topology graph, where $A_{ij}^t = 1$ if the ith node is attached to the jth node with an edge, and $A_{ij}^t = 0$ indicates that the edge is not connected. Meanwhile, $\mathbf{A}^f$ denotes the relationships of the feature graph, where $A_{ij}^f = 1$ if the ith node is attached to the jth node with an edge, and $A_{ij}^f = 0$ indicates that the edge is not connected. The objective of community detection is to partition the network into substructures that are tightly connected internally and sparsely connected externally. The notations used in this article are summarized in Table II.

IV. METHOD

Different from the previous works, in addition to considering the topology graph and feature embedding, FGCD proposes a new network embedding method that focuses on the feature graph, and incorporates it with topological structure and feature embedding for community detection. Fig. 2 depicts the overall architecture of FGCD. The method contains three modules, which are the AE module, GCN module, and GAE module. Specifically, the AE module is used to capture the representation of node features, the GCN module is used to capture the representation of the topology graph, and the GAE module is used to capture the representation of the feature graph. To fully integrate them, a layerwise fusion strategy is designed to utilize an attention mechanism. To facilitate joint optimization of community detection and network embedding during model training, a self-supervised mechanism is introduced to train the model. Each part of the model will be introduced in more detail as follows.

A. AE Module

The method FGCD is evaluated on 11 real networks with node features, which can be found in Table III. FGCD constructs an encoder to capture the representation $\mathbf{H}$ of node features. The multilayer perceptron (MLP) [46] is used to extract the embedding, where the lth layer is shown as follows:

$$\mathbf{H}_l = \text{ReLU}\left(\mathbf{W}_l^{\text{enc}}\mathbf{H}_{l-1} + \mathbf{b}_l^{\text{enc}}\right) \tag{1}$$

where $\mathbf{H}_l$ denotes the latent representation at the lth layer, and $\mathbf{W}_l^{\text{enc}}$ and $\mathbf{b}_l^{\text{enc}}$ represent the trained weights and biases in the lth layer, respectively. Moreover, $\mathbf{H}_0 = \mathbf{X}$ represents the original node features.

The decoder is also an MLP, whose lth layer is

$$\hat{\mathbf{H}}_l = \text{ReLU}\left(\mathbf{W}_l^{\text{dec}}\hat{\mathbf{H}}_{l-1} + \mathbf{b}_l^{\text{dec}}\right) \tag{2}$$

where $\hat{\mathbf{H}}_l$ represents the reconstructed node features at the lth layer, and $\mathbf{W}_l^{\text{dec}}$and $\mathbf{b}_l^{\text{dec}}$ the trained weights and biases of decoder lth layer, respectively. Besides, the output of the last layer (i.e., the Lth layer) $\hat{\mathbf{H}}_L$ is used as the reconstructed features $\hat{\mathbf{X}}$ of the original data. Then, the reconstruction loss of this AE module is applied as follows:

$$\mathcal{L}_{\text{ae}} = \|\mathbf{X} - \hat{\mathbf{X}}\|_F^2\,. \tag{3}$$

B. GCN Module

To capture the representation of the topology graph, the GCN module is applied to get a deeper representation $\mathbf{Z}^t$ of the topology graph. Specifically, the GCN module is constructed through a five-layer GCN and takes the topology graph $\mathbf{A}^t$ and node features $\mathbf{X}$ as input. The lth layer is shown as follows:

$$\mathbf{Z}_l^t = \text{ReLU}\left(\widetilde{\mathbf{D}}_t^{-\frac{1}{2}}\widetilde{\mathbf{A}}^t\widetilde{\mathbf{D}}_t^{-\frac{1}{2}}\mathbf{Z}_{l-1}^t\mathbf{W}_l^t\right) \tag{4}$$

where $\mathbf{Z}_l^t$ denotes the latent representation of the topology graph of the lth layer, $\widetilde{\mathbf{A}}^t = \mathbf{A}^t + \mathbf{I}$ and $\widetilde{\mathbf{D}}_t$ is the diagonal degree matrix of $\widetilde{\mathbf{A}}^t$. $\mathbf{W}_l^t$ is the network weight matrix of the topology graph at layer l of the GCN module.

C. GAE Module

To make the features of nodes propagate on the feature space, the feature graph is constructed G_f from the features of the nodes, where $\mathbf{A}_f$ indicates the adjacency matrix of the feature graph to represent the underlying relations of nodes in the feature space. Specifically, the similarity is calculated matrix $\mathbf{S}$ between different node features via the cosine similarity defined as follows:

$$\mathbf{S}_{ij} = \frac{\mathbf{x}_i^T\mathbf{x}_j}{|\mathbf{x}_i|\,|\mathbf{x}_j|} \tag{5}$$

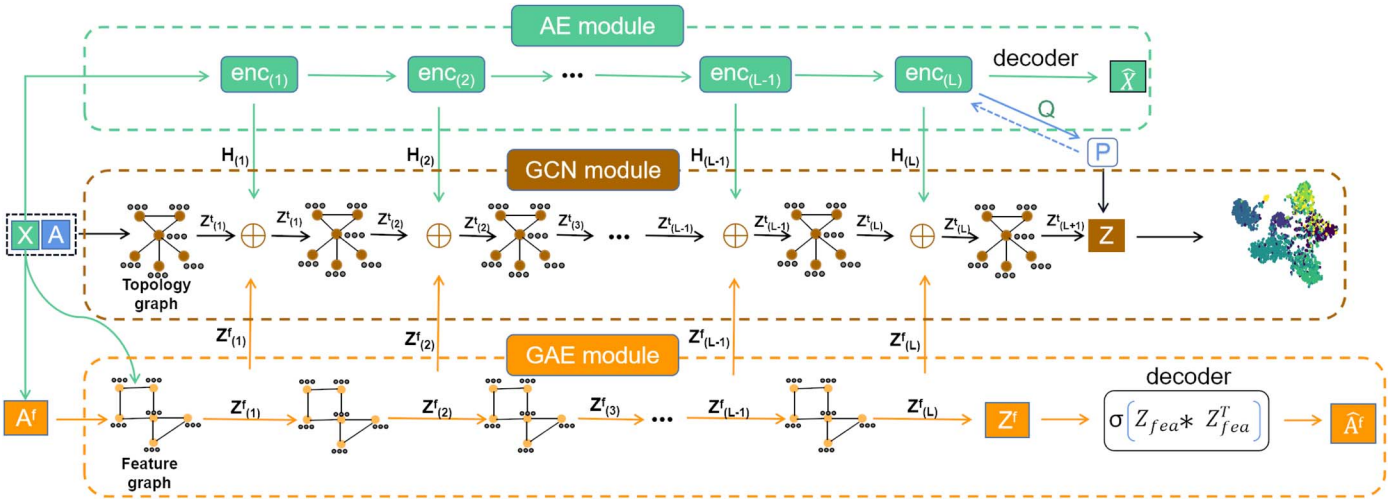

Fig. 2. FGCD framework. FGCD consists of the AE module, GCN module, and GAE module. $\mathbf{A}$ is the adjacency matrix of the topology graph. $\mathbf{A}^f$ is the adjacency matrix of the feature graph, and $\hat{\mathbf{A}}^f$ denotes the reconstructed adjacency matrix of the feature graph. $\mathbf{X}$ denotes the node features, and $\hat{\mathbf{X}}$ represents the reconstructed node features. L stands for the number of embedded layers, and $\oplus$ denotes the adaptive embedding fusion mechanism. The feature graph is constructed from $\mathbf{X}$. The embedding in the lth layer of the AE, GCN, and GAE modules are $\mathbf{H}_{(L)}$, $\mathbf{Z}^t_{(L)}$, and $\mathbf{Z}^f_{(L)}$, respectively.

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the feature vectors of the ith and jth nodes, respectively. Then, the k-nn graph is constructed on the similarity matrix $\mathbf{S}$ as the feature graph $\mathbf{G}_f$. After that, the representation of the feature graph is learned by utilizing the GAE module. GAE contains the graph convolutional encoder and decoder modules, which are shown as follows.

1) Graph Convolutional Encoder: First, a graph convolution encoder (g-encoder) is constructed. The g-encoder learns the potential representation $\mathbf{Z}^f$ via the GCN layer. The lth layer is shown as follows:

$$\mathbf{Z}^f_l = \mathrm{ReLU}\left(\widetilde{\mathbf{D}}_f^{-\frac{1}{2}}\widetilde{\mathbf{A}}^f\widetilde{\mathbf{D}}_f^{-\frac{1}{2}}\mathbf{Z}^f_{l-1}\mathbf{W}^f_l\right) \tag{6}$$

where $\widetilde{\mathbf{A}}^f = \mathbf{A}^f + \mathbf{I}$, $\widetilde{\mathbf{D}}_f$ is the diagonal degree matrix of $\widetilde{\mathbf{A}}^f$, and $\mathbf{W}^f_l$ is the network weight matrix of the feature graph at the lth layer. In particular, the g-encoder is constructed by a four-layer GCN. With this approach, the network representation $\mathbf{Z}^f$ of the feature graph can be acquired.

2) Graph Convolutional Decoder: Then, we use the GCD (g-decoder) to reconstruct the feature graph. Whether a connection exists between the two nodes is predicted by our g-decoder $p(\hat{\mathbf{A}}^f \mid \mathbf{Z}^f)$. Then, the GCD (g-decoder) is used to reconstruct the feature graph. Whether a connection exists between the two nodes is predicted by the g-decoder $p(\hat{\mathbf{A}}^f \mid \mathbf{Z}^f)$. Specially, g-decoder utilizes the node representation $\mathbf{Z}^f$ as input to obtain the reconstructed feature graph $\hat{\mathbf{A}}^f$. We define a link prediction layer based on g-decoder as follows:

$$p(\hat{\mathbf{A}}^f \mid \mathbf{Z}^f) = \prod_{i=1}^{n}\prod_{j=1}^{n} p\left(\hat{\mathbf{A}}^f_{ij} \mid \mathbf{z}^f_i, \mathbf{z}^f_j\right) \tag{7}$$

where

$$p\left(\hat{\mathbf{A}}^f_{ij} = 1 \mid \mathbf{z}^f_i, \mathbf{z}^f_j\right) = \mathrm{sigmoid}\left(\mathbf{z}_i^{f\top}, \mathbf{z}^f_j\right). \tag{8}$$

Therefore, the reconstructed adjacency matrix $\hat{\mathbf{A}}^f$ is obtained as follows:

$$\hat{\mathbf{A}}^f = \mathrm{sigmoid}\left(\mathbf{Z}^f\mathbf{Z}^{f\top}\right). \tag{9}$$

Then, the cross entropy loss is used to train the parameters of the GAE

$$\mathcal{L}_{\mathrm{gae}} = \mathbb{E}[\log p(\hat{\mathbf{A}}^f \mid \mathbf{Z}^f)]. \tag{10}$$

By (10), more reliable feature graph representations $\mathbf{Z}^f$ can be extracted to enhance the original GCN embeddings.

D. Adaptive Embedding Fusion Mechanism

Now, three distinct embeddings are available: $\mathbf{Z}^t$, $\mathbf{Z}^f$, and $\mathbf{H}$. Since these three embeddings are complementary to some extent, we propose a layerwise fusion mechanism to ensemble them more profoundly. In the process of the ensemble, an attention mechanism is employed to determine the significance of $\mathbf{Z}^t$, $\mathbf{Z}^f$, and $\mathbf{H}$ in each layer of embeddings. In more detail, $\mathbf{Z}^t_j$, $\mathbf{Z}^f_j$, and $\mathbf{H}_j$ are first concatenated to form $\left[\mathbf{Z}^t_j \| \mathbf{Z}^f_j \| \mathbf{H}_j\right] \in \mathbb{R}^{n\times 3d_j}$, where d_j is the dimensions of the embedding, to learn the related attention coefficients. The importance of the three module embeddings is then captured by introducing a fully connected layer parameterized by a weight matrix $\mathbf{W}_j \in \mathbb{R}^{3d_j\times 3}$. After that, the multiplication between $\left[\mathbf{Z}^t_j \| \mathbf{Z}^f_j \| \mathbf{H}_j\right]$ and $\mathbf{W}_j$ is fed into the Tanh activation function. Then, using the softmax function and ℓ_2 as follows:

$$\boldsymbol{\alpha}_j = \mathrm{Normalize}\left(\mathrm{softmax}\left(\mathrm{Tanh}\left(\left[\mathbf{Z}^t_j \| \mathbf{Z}^f_j \| \mathbf{H}_i\right]\mathbf{W}_j\right)\right)\right) \tag{11}$$

where $\mathrm{Normalize}(\cdot)$ is the ℓ_2 normalization. The attention coefficient $\boldsymbol{\alpha}_j$ can be rewritten as $\boldsymbol{\alpha}_j = \left[\boldsymbol{\alpha}^t_j \| \boldsymbol{\alpha}^f_j \| \boldsymbol{\alpha}^H_j\right]$, where $\boldsymbol{\alpha}^t_j$, $\boldsymbol{\alpha}^f_j$, and $\boldsymbol{\alpha}^H_j$ are weight vectors for quantifying the significance of $\mathbf{Z}^t_j$, $\mathbf{Z}^f_j$, and $\mathbf{H}_j$, respectively. Adaptively integrating the

embeddings of these three modules at the jth layer using the attention mechanism generates the network representation of the topology graph at the jth layer as follows:

$$\mathbf{Z}_j^t = \boldsymbol{\alpha}_j^t \cdot \mathbf{Z}_j^t + \boldsymbol{\alpha}_j^f \cdot \mathbf{Z}_j^f + \boldsymbol{\alpha}_j^H \cdot \mathbf{H}_j. \tag{12}$$

Then, the ensembled embedding shown in (12) is fed into the GCN module [i.e., (4)] to obtain the GCN representation of the next layer. The embedding result in the final layer is represented as $\mathbf{Z}$, which serves as the final embedding.

E. Objective Function

Since our task is community detection, after obtaining the final network embedding $\mathbf{Z}$, we adopt the BP algorithm [35] to calculate the loss of community detection. Since the task is community detection after obtaining the final network embedding $\mathbf{Z}$, the BP algorithm [35] is adopted to calculate the loss of community detection. The BP model, an improved variant of the BigCLAM [47] model, is regarded as a graph generation model, which can be used to detect community structures. The BP model equation is as follows:

$$A_{ij}^t \sim \text{Bernoulli}\left(1 - \exp\left(-\boldsymbol{Z}_i \boldsymbol{Z}_j^T\right)\right) \tag{13}$$

where $\boldsymbol{Z}_i$ and $\boldsymbol{Z}_j$ are the final embedding of the ith and jth nodes, respectively. Notice that when probability is used directly, the value is many small probabilities multiplied together and may lead to numerically unstable situations. Therefore, the negative log-likelihood of the BP model is used, which is shown as follows:

$$-\log p\left(\mathbf{A}^t|\mathbf{Z}\right) = -\sum_{(i,j)\in E} \log\left(1-\exp\left(-\boldsymbol{Z}_i \boldsymbol{Z}_j^T\right)\right) + \sum_{(i,j)\notin E} \boldsymbol{Z}_i \boldsymbol{Z}_j^T. \tag{14}$$

Because real-world graphs are typically sparse, the second part in (14) will contribute much more to the loss. To overcome this, balancing the two terms with the expectation is a common strategy for imbalanced classification [48]

$$\begin{aligned}\mathcal{L}_{\text{cd}} = &-\mathbb{E}_{(i,j)\sim P_{(i,j)\in E}}\left[\log\left(1-\exp\left(-\boldsymbol{Z}_i \boldsymbol{Z}_j^T\right)\right)\right] \\ &+ \mathbb{E}_{(i,j)\sim P_{(i,j)\notin E}}\left[\boldsymbol{Z}_i \boldsymbol{Z}_j^T\right].\end{aligned} \tag{15}$$

To develop a more reliable guide for community detection, we hope that the representation $\mathbf{H}$ of node features and the final embedding $\mathbf{Z}$ can be boosted by each other. The procedure is divided into two parts. To be more specific, the Student's t-distribution is utilized to quantify the similarity between the embedding point $\mathbf{h}_i$ and the center of cluster $\boldsymbol{\mu}_j$. Equation (16) is used to compute the probability q_{ij} that node i belongs to the jth community

$$q_{ij} = \frac{\left(1 + \left\|\mathbf{h}_i - \boldsymbol{\mu}_j\right\|^2 / v\right)^{-\frac{v+1}{2}}}{\sum_{j'}\left(1 + \left\|\mathbf{h}_i - \boldsymbol{\mu}_{j'}\right\|^2 / v\right)^{-\frac{v+1}{2}}} \tag{16}$$

where v is fixed to 1 in the implementation for simplicity. The probability that the ith sample belongs to the jth cluster center is computed as a target distribution P, as shown in (17), to boost the confidence in the cluster assignment. This can increase the compactness of the clusters, which brings samples closer to the cluster centers as follows:

$$p_{i,j} = \frac{q_{i,j}^2 / \sum_i q_{i,j}}{\sum_j' q_{i,j'}^2 / \sum_i q_{i,j'}}. \tag{17}$$

Since the target distribution P can be used to supervise the AE module and learn a better representation for the community detection task by minimizing the KL divergence. Unlike the traditional self-training mechanism, in addition to this, it is expected that P can guide the embedding $\mathbf{Z}$ in turn. Most of the work [18], [49] involves only one KL divergence loss. However, in the method, $\mathbf{Z}$ directly serves the downstream task of community detection, and a more reliable $\mathbf{Z}$ can lead to a better community detection result. To this end, another KL divergence term is added in

$$\begin{aligned}\mathcal{L}_{\text{KL}} &= \gamma_1 * \text{KL}(\mathbf{P}|\mathbf{Z}) + \gamma_2 * \text{KL}(\mathbf{P}|\mathbf{H}) \\ &= \gamma_1 \sum_i \sum_j p_{i,j} \log \frac{p_{i,j}}{z_{i,j}} + \gamma_2 \sum_i \sum_j p_{i,j} \log \frac{p_{i,j}}{q_{i,j}}\end{aligned} \tag{18}$$

where $\gamma_1, \gamma_2 > 0$ are the two hyperparameters. When approximating the distribution, KL divergence may be employed to reduce information loss, enabling the model to learn a wide variety of intricate distributions. Since both embedding $\mathbf{H}$ and $\mathbf{Z}$ are connected by the KL divergence with the P distribution, it establishes a strong connection between the AE module and the GCN module. By minimizing (18), the similar distribution of the AE module and the GCN module may be facilitated. To sum up, the overall objective function of FGCD contains four parts: community detection loss $\mathcal{L}_{\text{cd}}$, feature graph reconstruction loss $\mathcal{L}_{\text{gae}}$, KL divergence loss $\mathcal{L}_{\text{KL}}$, and topology graph reconstruction loss $\mathcal{L}_{\text{ae}}$. The final objective function is as follows:

$$\mathcal{L} = \mathcal{L}_{\text{cd}} + \mathcal{L}_{\text{KL}} + \beta * \mathcal{L}_{\text{ae}} + \theta * \mathcal{L}_{\text{gae}} \tag{19}$$

where β, θ are the balanced hyperparameters. The training process of the proposed FGCD is shown in Algorithm 1.

V. Experiments

A. Datasets

The method FGCD is evaluated on 11 real networks with node features. These datasets are widely used for the evaluation and comparison of related algorithms, such as those listed in the references [14], [19], [24]. The WebKB network consists of four subnetworks with 877 nodes and 1608 edges. Each subnetwork is divided into five communities and the attribute dimension of the nodes in each subnetwork is 1703. Wiki constitutes a network of webpages, wherein nodes represent individual webpages and are linked if one webpage refers to another. Within the Wiki structure, nodes are correlated with word vectors weighted by tf-idf. The Cora network (seven communities) consists of 2708 nodes and 5429 edges. The Citeseer network (six communities) consists of 3312 nodes and 4732 edges. The attribute dimensions of the nodes in Citeseer and Cora are 3703 and 1433, respectively. The ACM network (three communities) extracted from the ACM dataset consists

Algorithm 1: Training process of FGCD

Require: Feature matrix $\mathbf{X}$, adjacency matrix $\mathbf{A}$, number of communities k, iteration number I, hyperparameters β, θ, γ_1, γ_2;
Ensure: Final embedding $\mathbf{Z} \in \mathbb{R}^{n \times k}$;
1: Construct the adjacency matrix $\mathbf{A}^f$ from $\mathbf{X}$;
2: Initialize AE weights and biases by pre-training AE;
3: Initialize the clustering centers $\boldsymbol{\mu}$ with K-means on $\mathbf{H}$;
4: **while** $iter < I$ **do**
5: The embedding $\mathbf{H}_1$, ... , $\mathbf{H}_l$ of the features of the node is generated by Equation (1);
6: The embedding $\mathbf{Z}_1^f$, ... , $\mathbf{Z}_\ell^f$ of the feature graph is generated by Equation (6);
7: **for** $\ell \in 1, \ldots, l, (l+1)$ **do**
8: The embedding $\mathbf{Z}_1^t$, ... , $\mathbf{Z}_l^t$, $\mathbf{Z}_{l+1}^t$ of the topology graph is generated by Equation (4) and Equation (12);
9: **end for**
10: The final embedding $\mathbf{Z}$ is the $(l+1)$-th layer embedding of the topology graph;
11: Feed $\mathbf{H}_l$ into the AE's decoder to reconstruct the feature $\mathbf{X}$ of the node;
12: Feed $\mathbf{Z}_l^f$ into the GAE's decoder to reconstruct the feature graph $\mathbf{A}^f$;
13: Calculate $\mathcal{L}_{cd}$, $\mathcal{L}_{ae}$, $\mathcal{L}_{gae}$, $\mathcal{L}_{kl}$ respectively;
14: Calculate the loss function in Equation (19);
15: Update the parameters utilizing Adam Optimizer;
16: $iter$=$iter$+1;
17: **end while**
18: **return Z**

of 3025 nodes and 13 128 edges, where the nodes represent articles and if two articles have the same author, then there is an edge between them. The UAI2010 network (19 communities) consists of 3067 nodes and 28 311 edges which have been tested in GCNs for community detection. The large Cora network (ten communities) consists of 11 881 nodes and 64 898 edges which have more edges compared with other real datasets. The PubMed dataset network (three communities) consists of 19 717 nodes and 44 338 edges which forms a citation network for graph learning tasks.

B. Baselines and Evaluation Metrics

We compare FGCD with two kinds of community detection methods. One kind is the graph embedding method that only uses graph structures.

1) DeepWalk [26] uses a truncated sequence of random walks to learn the latent representation of local information.
2) LINE [28] is the general network embedding method that employs low-order proximity.
3) Node2vec [30] is an embedding method that is able to flexibly adjust the neighbors of a given node by adjusting the parameters.

TABLE III
ANALYTICAL OF 11 DATASETS, WHERE #NODES, #EDGES, #FEATURES, AND #CLASSES, RESPECTIVELY, DENOTE THE NUMBER OF NODES, EDGES, FEATURES, AND COMMUNITIES

Dataset	#Nodes	#Edges	#Features	#Classes
Cornell	195	286	1703	5
Texas	187	298	1703	5
Washington	230	417	1703	5
Wisconsin	265	479	1703	5
Wiki	2405	17 981	4973	17
Cora	2708	5429	1433	7
ACM	3025	13 128	1870	3
UAI2010	3067	28 311	4973	19
Citeseer	3312	4732	3703	6
Large Cora	11 881	64 898	3780	10
Pubmed	19 717	44 338	500	3

The other kind is the attributed graph clustering method which utilizes both node features and graph structures.

1) VGAE [50] migrates variational auto-encoders (VAEs) to the graph embedding.
2) AGC [51] proposes an adaptive graph convolution method and captures the global structure using higher order graph convolution.
3) SDCN [34] is the first to use structural information in the field of graph clustering.
4) FGC [52] presents a principled view of graph learning and fine-grained attribute graph clustering.
5) VGAER [25] utilizes variational reasoning techniques to model on topology graph and attribute information.
6) MA-GAE [53] enhances GAE and VGAE encoders by considering both the initial graph structure and modularity-based prior communities.
7) EGAE [54] proves that the relaxed k-means algorithm will obtain an optimal partition in the innerproduct distance metric space and utilizes the fact that the relaxed k-means and GAE can be learned simultaneously.

Five widely used metrics [55], namely accuracy (ACC), normalized mutual information (NMI), macro F1-score (F1), adjusted rand index (ARI), and modularity, are employed to assess the performance of community detection algorithms.

C. Experimental Setup and Implementation Details

The four hyperparameters γ_1, γ_2, β, and θ in the method are fixed as 0.01, 1000, 1000, and 1, respectively, on all datasets. For fair comparisons, the method references the network parameter settings as [34], the hidden layer dimension of the AE module is fixed to $500 \to 500 \to 2000 \to 10$, the hidden layer dimension of the GCN module is fixed to $500 \to 500 \to 2000 \to 10 \to k$ and the hidden layer dimension of the GAE module is set to $500 \to 500 \to 2000 \to 10$, where k represents the number of communities. There are two steps to the FGCD method's training procedure. To begin with, the AE module has been trained with 50 epochs, and its learning rate is fixed at 0.001. Following this, the number of training epochs for the entire model is set to 500. For the first, second, third, and fourth layers, ReLU activation and batch normalization are used. Adam is employed as the optimizer with a learning

TABLE IV
COMMUNITY DETECTION PERFORMANCE OF ALL COMPARISON ALGORITHMS IN TERMS OF FIVE EVALUATION METRICS ON 11 DATASETS

Dataset	Metrics	Deepwalk	LINE	Node2vec	VGAE	AGC	SDCN	FGC	VGAER	MA-GAE	EGAE	FGCD
Cornell	ACC	3.87e-1	4.54e-1	3.74e-1	3.74e-1	4.67e-1	4.11e-1	4.11e-1	4.41e-1	4.05e-1	3.59e-1	**4.75e-1**
	NMI	7.6e-2	9.4e-2	7.9e-2	7.6e-2	1.26e-1	1.45e-1	1.33e-1	1.03e-1	1.13e-1	1.24e-1	**2.28e-1**
	F1	2.45e-1	2.79e-1	2.6e-1	2.46e-1	3.34e-1	3.22e-1	2.52e-1	3.05e-1	3.37e-1	3.46e-1	**4.21e-1**
	ARI	1.3e-2	3.5e-2	6.4e-2	8.4e-2	1.12e-1	9.7e-2	4.9e-2	1.05e-1	6.7e-2	6.3e-2	**1.65e-1**
	Q	5.21e-1	2.1e-2	4.81e-1	5.17e-1	3.54e-1	2.43e-1	5.34e-1	5.12e-1	5.33e-1	5.08e-1	**5.41e-1**
Texas	ACC	4.07e-1	5.01e-1	3.73e-1	3.73e-1	4.86e-1	4.7e-1	4.70e-1	5.64e-1	4.65e-1	4.22e-1	**6.01e-1**
	NMI	5.6e-2	1.88e-1	1.82e-1	9.1e-2	1.96e-1	2.11e-1	1.35e-1	1.73e-1	8.9e-2	1.29e-1	**3.14e-1**
	F1	2.33e-1	3.04e-1	3.35e-1	3.09e-1	3.62e-1	3.63e-1	2.59e-1	3.66e-1	3.09e-1	3.09e-1	**4.63e-1**
	ARI	1.42e-1	1.76e-1	2.37e-1	1.79e-1	2.48e-1	1.61e-1	2.68e-1	2.25e-1	1.36e-1	1.2e-1	**3.12e-1**
	Q	3.97e-1	3.26e-1	3.44e-1	4.01e-1	3.61e-1	1.35e-1	4.62e-1	4.68e-1	**4.95e-1**	4.72e-1	4.75e-1
Washington	ACC	4.67e-1	5.33e-1	4.75e-1	4.75e-1	**5.56e-1**	4.44e-1	4.44e-1	5.04e-1	4.87e-1	4.09e-1	5.45e-1
	NMI	6.2e-2	1.82e-1	1.1e-1	8.9e-2	1.45e-1	1.22e-1	5.2e-2	1.37e-1	1.54e-1	1.3e-1	**2.73e-1**
	F1	2.25e-1	2.94e-1	2.86e-1	2.75e-1	2.95e-1	2.6e-1	1.79e-1	3.41e-1	3.17e-1	3.26e-1	**4.14e-1**
	ARI	5.1e-2	1.72e-1	2.27e-1	1.71e-1	2.33e-1	1.65e-1	9.2e-2	1.82e-1	1.47e-1	1.13e-1	**2.61e-1**
	Q	1.61e-1	1.28e-1	3.95e-1	3.79e-1	3.05e-1	2.36e-1	4.98e-1	4.94e-1	4.38e-1	**5.19e-1**	5.08e-1
Wisconsin	ACC	3.61e-1	3.94e-1	3.95e-1	3.95e-1	4.05e-1	4.35e-1	4.87e-1	4.37e-1	4.23e-1	3.47e-1	**5.17e-1**
	NMI	7.1e-2	9.2e-2	9.5e-2	9.2e-2	2.12e-1	1.63e-1	5.5e-1	1.31e-1	1.22e-1	1.25e-1	**2.33e-1**
	F1	2.83e-1	3.13e-1	2.68e-1	2.8e-1	3.35e-1	**4.09e-1**	1.8e-1	3.34e-1	3.22e-1	3.12e-1	3.98e-1
	ARI	6.6e-2	6.9e-2	6.2e-2	1.34e-1	1.32e-1	1.31e-1	9.2e-2	1.12e-1	4.3e-2	1.83e-1	**1.87e-1**
	Q	2.35e-1	7.9e-2	4.41e-1	4.96e-1	4.28e-1	4.22e-1	5.34e-1	5.22e-1	**6.02e-1**	5.41e-1	5.45e-1
Wiki	ACC	3.84e-1	2.77e-1	2.44e-1	2.44e-1	4.72e-1	3.9e-1	3.9e-1	4.07e-1	4.62e-1	4.72e-1	**5.61e-1**
	NMI	3.23e-1	2.12e-1	2.46e-1	3.02e-1	4.32e-1	3.5e-1	4.31e-1	3.58e-1	4.67e-1	4.51e-1	**5.14e-1**
	F1	2.57e-1	1.48e-1	2.21e-1	2.05e-1	3.87e-1	2.38e-1	3.42e-1	3.49e-1	4.03e-1	4.12e-1	**4.73e-1**
	ARI	1.82e-1	1.68e-1	1.62e-1	1.61e-1	2.41e-1	2.82e-1	1.79e-1	1.72e-1	2.49e-1	2.64e-1	**3.78e-1**
	Q	7.06e-1	6.89e-1	6.87e-1	5.93e-1	6.88e-1	6.2e-1	6.82e-1	6.03e-1	6.68e-1	7.26e-1	**7.29e-1**
Cora	ACC	4.51e-1	3.07e-1	5.57e-1	5.57e-1	6.72e-1	4.34e-1	4.34e-1	6.56e-1	6.34e-1	6.53e-1	**7.04e-1**
	NMI	3.15e-1	1.01e-1	4.01e-1	3.84e-1	5.28e-1	2.46e-1	**5.52e-1**	4.69e-1	5.08e-1	4.99e-1	5.37e-1
	F1	3.86e-1	2.31e-1	5.81e-1	4.15e-1	6.3e-1	3.71e-1	6.27e-1	6.48e-1	6.53e-1	6.27e-1	**6.86e-1**
	ARI	3.13e-1	1.58e-1	2.78e-1	4.48e-1	4.11e-1	2.56e-1	4.23e-1	3.91e-1	3.88e-1	4.42e-1	**4.78e-1**
	Q	7.08e-1	5.71e-1	6.64e-1	7.16e-1	7.13e-1	5.73e-1	6.74e-1	6.86e-1	7.17e-1	7.28e-1	**7.43e-1**
ACM	ACC	2.31e-1	1.97e-1	3.57e-1	4.39e-1	8.35e-1	**8.93e-1**	8.65e-1	5.07e-1	8.71e-1	8.26e-1	8.88e-1
	NMI	1.21e-1	1.37e-1	3.08e-1	1.13e-1	5.52e-1	6.51e-1	5.86e-1	2.11e-1	5.89e-1	4.88e-1	**6.86e-1**
	F1	3.59e-1	3.79e-1	3.29e-1	3.72e-1	8.36e-1	**8.93e-1**	5.86e-1	4.37e-1	8.71e-1	8.25e-1	8.89e-1
	ARI	2.25e-1	1.07e-1	1.04e-1	4.58e-1	5.78e-1	4.24e-1	6.44e-1	2.66e-1	6.54e-1	5.52e-1	**7.01e-1**
	Q	4.07e-1	3.26e-1	3.23e-1	6.21e-1	5.97e-1	5.85e-1	5.92e-1	6.03e-1	6.01e-1	5.33e-1	**6.16e-1**
UAI2010	ACC	3.45e-1	3.46e-1	3.44e-1	1.72e-1	3.45e-1	3.24e-1	3.76e-1	2.87e-1	2.63e-1	4.08e-1	**4.43e-1**
	NMI	3.11e-1	3.01e-1	3.17e-1	1.36e-1	3.43e-1	2.96e-1	3.25e-1	2.45e-1	2.28e-1	3.87e-1	**3.97e-1**
	F1	2.32e-1	2.57e-1	2.63e-1	1.48e-1	2.72e-1	2.53e-1	2.36e-1	2.41e-1	2.4e-1	3.58e-1	**3.61e-1**
	ARI	1.17e-1	1.54e-1	1.41e-1	3.2e-2	2.03e-1	1.51e-1	1.32e-1	1.12e-1	8.5e-2	2.11e-1	**2.33e-1**
	Q	3.41e-1	3.58e-1	3.18e-1	5.8e-2	4.06e-1	3.51e-1	4.12e-1	2.53e-1	2.23e-1	3.93e-1	**4.18e-1**
Citeseer	ACC	3.62e-1	2.51e-1	3.6e-1	3.6e-1	6.64e-1	6.35e-1	6.35e-1	4.55e-1	6.15e-1	6.22e-1	**6.78e-1**
	NMI	1.06e-1	5.6e-2	1.69e-1	2.27e-1	4.05e-1	3.71e-1	**4.31e-1**	2.49e-1	3.55e-1	3.51e-1	4.26e-1
	F1	3.42e-1	1.58e-1	3.18e-1	3.18e-1	6.18e-1	5.79e-1	6.34e-1	4.42e-1	5.85e-1	5.91e-1	**6.45e-1**
	ARI	1.73e-1	3.4e-2	7.1e-2	2.82e-1	4.01e-1	3.36e-1	4.12e-1	1.53e-1	3.48e-1	3.55e-1	**4.38e-1**
	Q	7.23e-1	4.42e-1	7.16e-1	7.22e-1	6.94e-1	7.09e-1	7.13e-1	7.12e-1	**7.66e-1**	7.51e-1	7.55e-1
Large Cora	ACC	4.17e-1	2.59e-1	3.21e-1	3.21e-1	4.23e-1	4.13e-1	**4.71e-1**	3.48e-1	3.53e-1	3.95e-1	4.31e-1
	NMI	2.43e-1	1.06e-1	2.32e-1	2.88e-1	3.05e-1	1.73e-1	3.01e-1	2.71e-1	2.73e-1	3.06e-1	**3.39e-1**
	F1	2.16e-1	1.96e-1	2.37e-1	2.84e-1	3.23e-1	1.69e-1	2.21e-1	3.48e-1	2.93e-1	3.56e-1	**3.92e-1**
	ARI	1.75e-1	6.8e-2	6.9e-2	1.66e-1	1.24e-1	**2.38e-1**	1.76e-1	1.31e-1	1.59e-1	1.77e-1	2.24e-1
	Q	6.77e-1	5.95e-1	6.66e-1	6.77e-1	6.42e-1	4.72e-1	7.18e-1	5.38e-1	5.78e-1	7.57e-1	**7.63e-1**
Pubmed	ACC	6.43e-1	4.83e-1	6.32e-1	5.85e-1	6.21e-1	6.64e-1	6.54e-1	5.43e-1	5.29e-1	**6.92e-1**	6.71e-1
	NMI	2.84e-1	1.44e-1	2.83e-1	2.46e-1	2.97e-1	2.67e-1	**3.04e-1**	1.51e-1	1.64e-1	2.95e-1	2.74e-1
	F1	6.26e-1	4.88e-1	6.28e-1	5.79e-1	6.14e-1	6.67e-1	6.54e-1	4.97e-1	4.82e-1	**6.83e-1**	6.73e-1
	ARI	2.15e-1	6.3e-2	2.43e-1	2.18e-1	2.46e-1	2.65e-1	2.72e-1	1.16e-1	1.74e-1	**3.06e-1**	2.74e-1
	Q	5.86e-1	4.45e-1	5.82e-1	5.74e-1	4.71e-1	5.42e-1	5.64e-1	5.33e-1	2.54e-1	6.01e-1	**6.53e-1**

Note: Bold—The best; Underline—The second best.

rate of 0.001 on all benchmark datasets. In particular, for both DeepWalk and Node2vec, the window size, the walk length, and the number of walks per node are set to 10, 80, and 10, respectively. For LINE, the settings of parameters are followed as in [28]. The embedding vectors are normalized by the L2-norm. For VGAE, encoders are designed with a 32-neuron hidden layer followed by a 16-neuron embedding layer. The training process spans 200 epochs and employs the Adam optimizer with a learning rate of 0.01. For FGCD, AGC, FGC, VGAER, MA-GAE, and EGAE, to ensure the consistency of evaluation results, we ran the source code of the comparative methods on all datasets.

D. Result Analysis

For each dataset, we run each method ten times and select the best result each time. The average of the ten best results is reported in Table IV. The top and second best results are bold and underlined, respectively.

Fig. 3. Visualization of the network embedding on the Cora dataset. (a) VGAE. (b) VGAER. (c) MA-GAE. (d) FGCD.

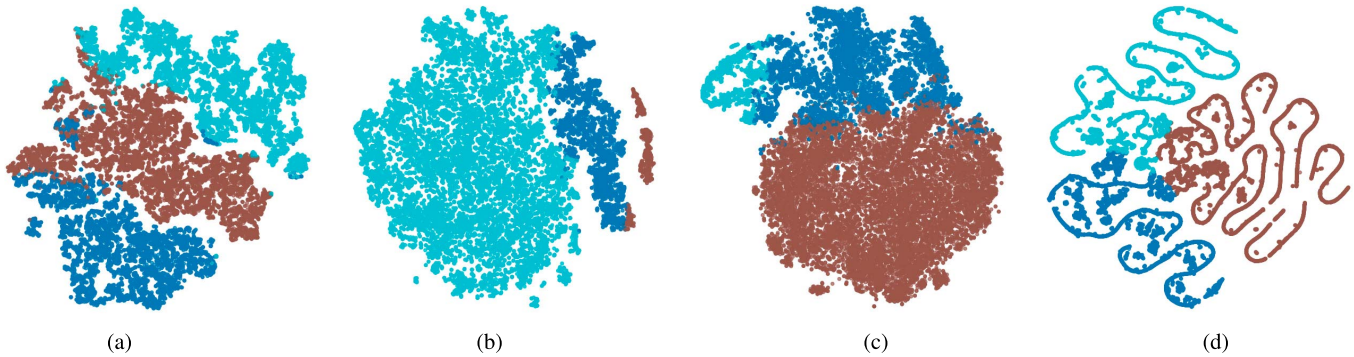

Fig. 4. Visualization of the network embedding on the Pubmed dataset. (a) VGAE. (b) VGAER. (c) MA-GAE. (d) FGCD.

On most datasets, the method yields the best community detection performance. With regard to the remaining datasets, the proposed one can achieve the second-best results. For example, compared with the state-of-the-art method (i.e., EGAE) in the ACM dataset, the proposed FGCD achieves 8.94%, 9.04%, 25.71%, 28.86%, and 12.48% improvements on ACC, NMI, F1, ARI, and Q, respectively. It well demonstrates the effectiveness of the method by considering feature graph fusion.

FGCD significantly outperforms network embedding-based methods. This is due to the fact that FGCD also utilizes node features and the feature graph, which can considerably improve community detection performance. By fusing the representations of the three modules and fully exploiting the information most relevant to the node labels, FGCD can easily achieve better performance.

FGCD can also outperform the attribute community detection methods. This is because FGCD also focuses on the feature graph, which can better characterize the relations between nodes in the feature space, whereas conventional embeddings cannot effectively capture such relations. Moreover, VGAE, AGC, FGC, SDCN, VGAER, MA-GAE, and EGAE treat the significance of topology graph and node features equally. However, in practice, they have different importance and should not be handled equally. In the method, the embedding weights of these three modules are learned by the attention mechanism, which guarantees the appropriate weight assignment.

E. Visualization

To further demonstrate that the representation of FGCD is reliable, the representation of FGCD and the representation of the other methods (i.e., VGAE, VGAER, and MA-GAE) are visualized on two representative datasets Cora and Pubmed. The representation is reduced to two dimensions using the t-SNE [56] tool, and each category label is assigned a color. Figs. 3 and 4 show the results on the Cora and Pubmed datasets, respectively. From these figures, we can see that the representations obtained by VGAE, VGAER, and MA-GAE cannot clearly show the structure of the community. Moreover, the embedding results in Figs. 3(d) and 4(d) show that FGCD partitions the nodes in different communities more clearly. Besides, most nodes of the same color are grouped together, indicating that the model can get high-quality embedding.

F. Ablation Experiment

In this section, a series of ablation experiments are performed to verify the validity of the method. We consider three variants of FGCD to verify the effectiveness of the method on all datasets.

1) FGCD_wo_1: obtains the network embedding without considering node features and feature graph as inputs.
2) FGCD_wo_2: obtains the network embedding without considering the node features as input.

TABLE V
COMPARISON BETWEEN FGCD AND ITS VARIANTS IN TERMS OF FIVE EVALUATION METRICS ON 11 DATASETS

Metrics	Methods	Cornell	Texas	Washington	Wisconsin	Wiki	Cora	ACM	UAI2010	Citeseer	Large Cora	Pubmed
ACC	FGCD_ wo_1	4.34e-1	4.43e-1	4.86e-1	4.17e-1	4.22e-1	6.04e-1	7.16e-1	3.22e-1	4.93e-1	3.76e-1	5.77e-1
	FGCD_wo_2	4.49e-1	4.71e-1	4.34e-1	4.18e-1	4.62e-1	5.91e-1	8.53e-1	3.37e-1	6.15e-1	4.13e-1	6.46e-1
	FGCD_wo_3	3.91e-1	4.17e-1	4.24e-1	3.81e-1	4.22e-1	6.21e-1	7.73e-1	3.12e-1	5.92e-1	4.23e-1	6.52e-1
	FGCD_wo_e2e	4.31e-1	3.82e-1	4.04e-1	3.72e-1	3.36e-1	3.45e-1	5.75e-1	2.71e-1	3.61e-1	2.45e-1	4.39e-1
	FGCD	**4.75e-1**	**6.01e-1**	**5.45e-1**	**5.17e-1**	**5.61e-1**	**7.04e-1**	**8.88e-1**	**4.43e-1**	**6.78e-1**	**4.31e-1**	**6.71e-1**
NMI	FGCD_wo_1	1.61e-1	1.44e-1	1.55e-1	1.31e-1	3.83e-1	4.65e-1	3.54e-1	3.15e-1	2.35e-1	2.77e-1	2.03e-1
	FGCD_wo_2	2.18e-1	1.71e-1	2.06e-1	1.88e-1	4.26e-1	4.95e-1	5.68e-1	3.35e-1	3.89e-1	3.05e-1	2.57e-1
	FGCD_wo_3	1.16e-1	1.46e-1	1.62e-1	1.41e-1	3.75e-1	4.76e-1	4.07e-1	2.63e-1	2.97e-1	2.86e-1	2.54e-1
	FGCD_wo_e2e	1.73e-1	1.34e-1	1.65e-1	1.65e-1	2.77e-1	1.47e-1	2.05e-1	2.36e-1	1.29e-1	8.3e-2	9.4e-2
	FGCD	**2.28e-1**	**3.14e-1**	**2.73e-1**	**2.33e-1**	**5.14e-1**	**5.37e-1**	**6.86e-1**	**3.97e-1**	**4.26e-1**	**3.39e-1**	**2.74e-1**
F1	FGCD_wo_1	3.31e-1	2.95e-1	3.37e-1	3.43e-1	3.15e-1	5.56e-1	7.21e-1	2.86e-1	4.13e-1	3.15e-1	5.47e-1
	FGCD_wo_2	4.12e-1	3.36e-1	3.28e-1	3.31e-1	3.97e-1	5.63e-1	8.31e-1	2.79e-1	5.62e-1	3.14e-1	6.24e-1
	FGCD_wo_3	3.22e-1	3.28e-1	3.16e-1	3.21e-1	3.46e-1	5.25e-1	7.72e-1	2.14e-1	5.36e-1	2.99e-1	5.93e-1
	FGCD_wo_e2e	3.36e-1	3.23e-1	3.22e-1	2.99e-1	2.75e-1	3.21e-1	5.43e-1	2.57e-1	3.46e-1	1.83e-1	4.36e-1
	FGCD	**4.21e-1**	**4.63e-1**	**4.14e-1**	**3.98e-1**	**4.73e-1**	**6.86e-1**	**8.89e-1**	**3.61e-1**	**6.45e-1**	**3.92e-1**	**6.73e-1**
ARI	FGCD_wo_1	7.12e-2	1.08e-1	1.03e-1	9.3e-2	1.95e-1	2.06e-1	3.21e-1	1.86e-1	3.13e-1	1.21e-1	2.03e-1
	FGCD_wo_2	1.02e-1	1.86e-1	1.52e-1	1.01e-1	2.81e-1	3.71e-1	6.05e-1	2.08e-1	3.74e-1	1.84e-1	2.45e-1
	FGCD_wo_3	6.2e-2	5.2e-2	9.6e-2	6.9e-2	3.66e-1	3.45e-1	6.72e-1	2.01e-1	2.56e-1	1.99e-1	2.28e-1
	FGCD_wo_e2e	1.49e-1	7.8e-2	1.13e-1	7.8e-2	1.32e-1	1.22e-1	1.95e-1	1.12e-1	1.22e-1	4.5e-2	7.9e-2
	FGCD	**1.65e-1**	**3.12e-1**	**2.61e-1**	**1.87e-1**	**3.78e-1**	**4.78e-1**	**7.01e-1**	**2.33e-1**	**4.38e-1**	**2.24e-1**	**2.74e-1**
Q	FGCD_wo_1	5.03e-1	3.85e-1	4.58e-1	4.63e-1	7.02e-1	7.16e-1	5.71e-1	3.68e-1	6.73e-1	6.57e-1	5.84e-1
	FGCD_wo_2	5.21e-1	4.06e-1	4.79e-1	5.17e-1	7.26e-1	7.34e-1	5.84e-1	4.12e-1	7.45e-1	6.87e-1	6.18e-1
	FGCD_wo_3	5.12e-1	4.58e-1	4.76e-1	4.71e-1	7.16e-1	7.25e-1	4.89e-1	3.64e-1	7.16e-1	6.99e-1	6.07e-1
	FGCD_wo_e2e	3.89e-1	3.56e-1	3.75e-1	3.82e-1	5.85e-1	4.49e-1	5.12e-1	3.02e-1	5.64e-1	4.86e-1	3.98e-1
	FGCD	**5.41e-1**	**4.75e-1**	**5.08e-1**	**5.45e-1**	**7.29e-1**	**7.43e-1**	**6.16e-1**	**4.18e-1**	**7.55e-1**	**7.63e-1**	**6.53e-1**

Note: Bold—The best.

3) FGCD_wo_3: obtains the network embedding without considering the feature graph as an input.
4) FGCD_wo_e2e: obtains the network embedding without performing the end-to-end optimization.

In other words, the ablation study consists of two parts. The first part aims to investigate the impact of feature graphs, node features, and their combination on the embedding. Specifically, an experiment was conducted to determine whether the inclusion or exclusion feature graphs or node features, or both together, affect the quality of the embedding. The second part aims to verify the effectiveness of the end-to-end optimization method by directly performing K-means clustering on the learned representations. Table V shows the comparison results of FGCD and its variants. From Table V, some conclusions can be drawn as follows.

1) Comparing the results of FGCD_wo_1 with FGCD_wo_2 and FGCD_wo_3, in general, the results of FGCD_wo_2 and FGCD_wo_3 are better than FGCD_wo_1, which indicates the effectiveness of node features and feature graph.
2) The results of FGCD_wo_2 are better than those of FGCD_wo_3 under most datasets, which indicates that the feature graph contains better information than the node features.
3) The results of FGCD significantly outperform those of FGCD_wo_e2e on all datasets, demonstrating the importance of end-to-end optimization for deep community detection tasks.
4) Overall, the embedding quality of FGCD is higher than that of its four variants, which shows the effectiveness of fully combining topology graph, node features, and feature graph.

VI. CONCLUSION

In community detection, to better facilitate the node feature propagation in both feature space and topology space, a new method FGCD is proposed, which constructs a feature graph using attribute information of nodes to characterize the relations between nodes in the feature space. In FGCD, a layerwise fusion mechanism is designed to ensemble the node features, topology graph, and feature graph seamlessly. In addition, to more appropriately assign the embedding weights of each part, an attention mechanism is suggested in each layer of feature fusion. We conduct extensive experiments by comparing the proposed FGCD with the state-of-the-art methods on a variety of real-world datasets. The experimental results have shown the proposed method outperforms the baseline methods on the majority of datasets, which well demonstrates the superiority and effectiveness of FGCD. In the intricate task of community detection, the automatic determination of the number of communities has long been a vexing issue. Moving forward, in the future, we will tackle this challenge by introducing the concept of pseudolabels in FGCD.

REFERENCES

[1] B. S. Khan and M. A. Niazi, "Network community detection: A review and visual survey," 2017, *arXiv:1708.00977*.
[2] P. Chunaev, "Community detection in node-attributed social networks: A survey," *Comput. Sci. Rev.*, vol. 37, 2020, Art. no. 100286.
[3] F. Liu et al., "Deep learning for community detection: Progress, challenges and opportunities," in *Proc. Int. Joint Conf. Artif. Intell.*, 2020, pp. 4981–4987.
[4] N. J. Krogan et al., "Global landscape of protein complexes in the yeast saccharomyces cerevisiae," *Nature*, vol. 440, no. 7084, pp. 637–643, 2006.
[5] M. E. Newman, "Detecting community structure in networks," *Eur. Phys. J. B*, vol. 38, no. 2, pp. 321–330, 2004.

[6] F. Wang, T. Li, X. Wang, S. Zhu, and C. H. Q. Ding, "Community discovery using nonnegative matrix factorization," *Data Mining Knowl. Discovery*, vol. 22, no. 3, pp. 493–521, 2011.

[7] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2013, pp. 587–596.

[8] Y. Jia, Q. Zhang, W. Zhang, and X. Wang, "CommunityGAN: Community detection with generative adversarial nets," in *Proc. World Wide Web Conf.*, 2019, pp. 784–794.

[9] F. Sun, M. Qu, J. Hoffmann, C. Huang, and J. Tang, "vGraph: A generative model for joint community detection and node representation learning," 2019, pp. 512–522.

[10] J. Chen et al., "Self-training enhanced: Network embedding and overlapping community detection with adversarial learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6737–6748, Nov. 2022.

[11] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity based community detection with deep learning." in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 2252–2258.

[12] S. Cavallari, V. W. Zheng, H. Cai, K. C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 377–386.

[13] F. Ye, C. Chen, and Z. Zheng, "Deep autoencoder-like nonnegative matrix factorization for community detection," in *Proc. Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 1393–1402.

[14] X. Wang, D. Jin, X. Cao, L. Yang, and W. Zhang, "Semantic community identification in large attribute networks," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 265–271.

[15] D. He, Z. Feng, D. Jin, X. Wang, and W. Zhang, "Joint identification of network communities and semantics via integrative modeling of network topologies and node contents," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 116–124.

[16] X. Xu, Y. Xiao, X. Yang, L. Wang, and Y. Zhou, "Attributed network community detection based on network embedding and parameter-free clustering," *Appl. Intell.*, vol. 52, no. 7, pp. 8073–8086, 2022.

[17] Y. Zhang et al., "SEAL: Learning heuristics for community detection with generative adversarial networks," in *Proc. Conf. Knowl. Discovery Data Mining*, 2020, pp. 1103–1113.

[18] H. Sun et al., "Network embedding for community detection in attributed networks," *ACM Trans. Knowl. Discovery Data*, vol. 14, no. 3, pp. 1–25, 2020.

[19] Y. Li, C. Sha, X. Huang, and Y. Zhang, "Community detection in attributed graphs: An embedding approach," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 338–345.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[21] X. Yang, C. Deng, Z. Dang, K. Wei, and J. Yan, "SelfSAGCN: Self-supervised semantic alignment for graph convolution network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 16775–16784.

[22] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 639–648.

[23] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei, "AM-GCN: Adaptive multi-channel graph convolutional networks," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2020, pp. 1243–1253.

[24] D. He et al., "Community-centric graph convolutional network for unsupervised community detection," in *Proc. Int. Joint Conf. Artif. Intell.*, 2021, pp. 3515–3521.

[25] C. Qiu, Z. Huang, W. Xu, and H. Li, "VGAER: Graph neural network reconstruction based community detection," 2022, *arXiv:2201.04066*.

[26] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 701–710.

[27] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proc. AAAI Conf. Artif. Intell.*, 2014, vol. 28, no. 1.

[28] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. Int. Conf. World Wide Web*, pp. 1067–1077, 2015.

[29] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1225–1234.

[30] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.

[31] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, no. 1, 2016, pp. 1145–1152.

[32] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 203–209.

[33] J. Zhu, C. Wang, C. Gao, F. Zhang, Z. Wang, and X. Li, "Community detection in graph: An embedding method," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 2, pp. 689–702, Mar./Apr. 2022.

[34] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, "Structural deep clustering network," in *Proc. Web Conf.*, 2020, pp. 1400–1410.

[35] O. Shchur and S. Günnemann, "Overlapping community detection with graph neural networks," 2019, *arXiv:1909.12201*.

[36] K. Berahmand, M. Mohammadi, F. Saberi-Movahed, Y. Li, and Y. Xu, "Graph regularized nonnegative matrix factorization for community detection in attributed networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 1, pp. 372–385, Jan./Feb. 2023.

[37] N. Mrabah, M. Bouguessa, M. F. Touati, and R. Ksantini, "Rethinking graph auto-encoder models for attributed graph clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9037–9053, Sep. 2023.

[38] D.-D. Lu, J. Qi, J. Yan, and Z.-Y. Zhang, "Community detection combining topology and attribute information," *Knowl. Inf. Syst.*, vol. 64, no. 2, pp. 537–558, 2022.

[39] L. Gong, S. Zhou, X. Liu, and W. Tu, "Attributed graph clustering with dual redundancy reduction," in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 3015–3021.

[40] B. F. Kamhoua, L. Zhang, K. Ma, J. Cheng, B. Li, and B. Han, "Grace: A general graph convolution framework for attributed graph clustering," *ACM Trans. Knowl. Discovery Data*, vol. 17, no. 3, pp. 1–31, 2023.

[41] R. Shang, W. Zhang, Z. Li, C. Wang, and L. Jiao, "Attribute community detection based on latent representation learning and graph regularized non-negative matrix factorization," *Appl. Soft Comput.*, vol. 133, 2023, Art. no. 109932.

[42] K. Berahmand, Y. Li, and Y. Xu, "A deep semi-supervised community detection based on point-wise mutual information," *IEEE Trans. Comput. Social Syst.*, early access, Nov. 8, 2023.

[43] K. Berahmand, M. Mohammadi, R. Sheikhpour, Y. Li, and Y. Xu, "WSNMF: Weighted symmetric nonnegative matrix factorization for attributed graph clustering," *Neurocomputing*, vol. 566, 2024, Art. no. 127041.

[44] D. Jin, Z. Yu, P. Jiao, S. Pan, P. S. Yu, and W. Zhang, "A survey of community detection approaches: From statistical modeling to deep learning," *IEEE Trans. Knowl. Data Eng.*, vol. 35, pp. 1149–1170, Aug. 2021.

[45] X. Su et al., "A comprehensive survey on community detection with deep learning," 2021, Art. no. 2105.12584.

[46] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—A review of applications in the atmospheric sciences," *Atmospheric Environ.*, vol. 32, nos. 14–15, pp. 2627–2636, 1998.

[47] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2013, pp. 587–596.

[48] S. Kotsiantis et al., "Handling imbalanced datasets: A review," *GESTS Int. Trans. Comput. Sci. Eng.*, vol. 30, no. 1, pp. 25–36, 2006.

[49] J. Xie, R. B. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. Mach. Learn.*, vol. 48, 2016, pp. 478–487.

[50] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. NIPS Workshop Bayesian Deep Learn.*, 2016.

[51] X. Zhang, H. Liu, Q. Li, and X.-M. Wu, "Attributed graph clustering via adaptive graph convolution," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 4327–4333.

[52] Z. Kang, Z. Liu, S. Pan, and L. Tian, "Fine-grained attributed graph clustering," in *Proc. Int. Conf. Data Mining*, 2022, pp. 370–378.

[53] G. Salha-Galvan, J. F. Lutzeyer, G. Dasoulas, R. Hennequin, and M. Vazirgiannis, "Modularity-aware graph autoencoders for joint community detection and link prediction," *Neural Netw.*, vol. 153, pp. 474–495, 2022, doi: 10.1016/J.NEUNET.2022.06.021.

[54] H. Zhang, P. Li, R. Zhang, and X. Li, "Embedding graph auto-encoder for graph clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 9352–9362, Nov. 2023.

[55] C. C. Aggarwal and C. K. Reddy, *Data Clustering: Algorithms and Applications*. Boca Raton, FL, USA: CRC Press, 2014.

[56] L. van der Maaten and G. E. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008.

Lei Zhang received the B.Sc. degree from Anhui Agriculture University, Hefei, China, and the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2007 and 2014, respectively, all in computer science and technology.

Currently, he is an Associate Professor with the School of Computer Science and Technology, Anhui University, Hefei, China. His research interests include multiobjective optimization and their applications, data mining, machine learning, and social network analysis and recommendation. He has published more than 90 papers in refereed journals and conferences, such as IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON BIG DATA, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, *ACM TKDD*, IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE, *Information Sciences*, *AAAI*, *IJCAI*, and *ACM SIGKDD*.

Dr. Zhang was the recipient of the ACM CIKM'12 Best Student Paper Award. He is a member of ACM.

Zeqi Wu received the B.Sc. degree from the School of Computer Science and Technology, Jinggangshan University, China, in 2021. He is currently working toward the master's degree with the School of Computer Science and Technology, Anhui University, Hefei, China.

His research interests include graph embedding and community detection.

Haipeng Yang received the B.Sc. degree from the School of Computer Science and Technology, Anhui University, Hefei, China, in 2019, where he is currently working toward the Ph.D. degree.

His research interests include multiobjective optimization and social network analysis.

Wuji Zhang received the B.Sc. degree from the School of Computer Science and Technology, Anhui Agricultural University of Technology, China, in 2021. He is currently working toward the master's degree with the School of Computer Science and Technology, Anhui University, Hefei, China.

His research interests include graph neural network, social recommendation system, and multi-behavior recommendation system.

Peng Zhou (Senior Member, IEEE) received the B.Sc. degree in computer science and technology from the University of Science and Technology of China, in 2011, and the Ph.D. degree in computer science from the Institute of Software, University of Chinese Academy of Sciences, Beijing, China, in 2017.

He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University, Hefei, China. His research interests include machine learning, data mining, and artificial intelligence. He has published more than 40 papers in highly regarded conferences and journals, including IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, *ACM TKDD*, *Pattern Recognition*, *IJCAI*, *AAAI*, *ACM MM*, *SDM*, and *ICDM*.