Graph Diffusion Enhanced Contrastive Representation Learning on Topology and Feature Space for Community Detection

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

***Abstract*—Community detection is a fundamental task for understanding complex networks. Many traditional community detection methods only focus on the relationships between nodes in the topology space (i.e., the topology graph) and node attributes. These methods do not fully consider the relationships between nodes in the feature space (i.e., the feature graph) and the global structure of the graph. To this end, we introduce an innovative approach termed Graph Diffusion enhanced Contrastive representation Learning (GDCL) method on topology and feature space for community detection. Specifically, we first construct a feature graph from node attributes to capture the relationships between nodes in the feature space. Based on the topology graph and feature graph, we deploy the graph diffusion module to generate topology diffusion and feature diffusion graphs respectively. Then, we leverage the shared graph convolutional and graph contrastive learning modules to learn node representations of the local and global structures of the graphs. Simultaneously, we employ the hidden layer adaptive embedding fusion mechanism to obtain high-quality node representations. Finally, the embedding integration strategy integrates them with feature embedding to obtain the final embedding for community detection. Extensive experiments on ten real-world datasets demonstrate that GDCL outperforms the existing state-of-the-art algorithms.**

***Index Terms*—Community detection, contrastive learning, attributed networks, feature graph.**

I. Introduction

IN COMPLEX systems, nodes can form different communities through their mutual connections. Community detection [1], [2], [3] is a task to detect communities by partitioning the nodes in a graph into clusters, which helps to understand complex systems. Therefore, community detection is a fundamental problem in graph analysis and has been widely used in many real-world applications, such as bioinformatics [4], social network analysis [5], and transportation networks [6].

Community detection typically considers two aspects of information: the relationships between nodes in the topology space (i.e., the topology graph) and the node attributes. Traditional community detection research [7], [8], [9], [10], [11] has mainly focused on analyzing the topology structure of graphs. For example, Pan et al. [12] pioneers the application of deep learning to community detection. Ye et al. [13] combines autoencoders and non-negative matrix factorization (NMF) for complex network analysis. Some other research has focused on feature embedding and its integration. As an example, the work of Li et al. [14] presents an approach to community structure embedding that is applicable to attributed graphs. Recently, He et al. [15] proposes a new unsupervised community detection method for GCN-based attribute networks.

In addition, the recent work by [16] explores the theoretical advancements, new models, algorithms, and various applications in the emerging field of Deep Neural Network Graphs (DNNG), highlighting their potential in community detection and related domains.

Despite the promising performance of the aforementioned studies on the community detection task, it can be observed that they have two main limitations. Firstly, they ignore the node relations in the feature space (i.e., feature graph). In fact, the feature graph can compensate for the information that the topology graph ignores. However, existing methods often independently process nodes within the feature space, ignoring the relationships of nodes in the feature space. For example, Fig. 1(a) shows the true network partition, where nodes A, B, C, and F belong to one community, nodes D and E belong to one community, and nodes G, H, and I belong to one community. Fig. 1(b), (c), and (d) show the result of only using the topology graph, attribute information, and feature graph, respectively. When considering nodes D and E, we find that neither the topology graph-based result nor the attribute information-based result can put them into the same community. However, in the feature graph-based result, they are correctly assigned to the same community. It shows that the feature graph contains some compensation information for the topology graph and attribute information, which can enhance the conventional methods which only use the topology graph and attribute information.

Received 27 May 2024; revised 8 January 2025; accepted 24 May 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 61976001and Grant 62176001, in part by the Natural Science Foundation of Anhui Province under Grant 2408085MF152, 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, School of Computer Science and Technology, Anhui University, Hefei 230039, China (e-mail: zl@ahu.edu.cn; e24201024@stu.ahu.edu.cn; a2467449838@163.com; e21201077@stu.ahu.edu.cn; haipengyang@126.com; zhoupeng@ahu.edu.cn).

Our code is available at https://github.com/vege12138/GDCL.

This article has supplementary downloadable material available at https://doi.org/10.1109/TETCI.2025.3577494, provided by the authors.

Recommended for acceptance by D. Camacho.

Digital Object Identifier 10.1109/TETCI.2025.3577494

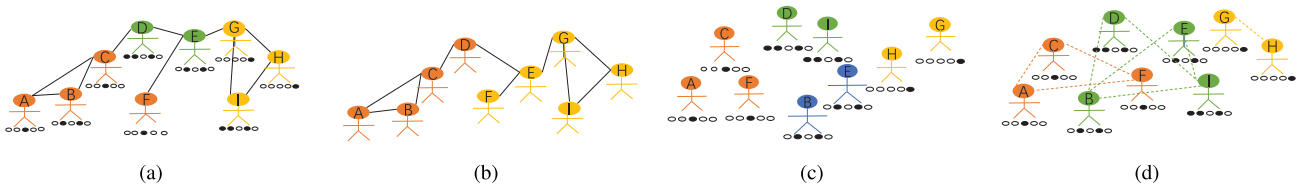


Fig. 1. (a) the division of the original network. (b) the topology graph-based network partitioning. (c) the attribute information-based network partitioning. (d) the feature graph-based network partitioning.

Secondly, they fail to comprehensively consider the global information of the graph. The conventional methods only focus on the local graph structure of the nearest neighbor nodes but fail to fully explore the global structure of the graph, leading to information loss and unstable embedding problems. It is worth noting that a few studies [17], [18], [19] in supervised learning have leveraged graph diffusion to explore global information. Although these methods can alleviate this problem to some extent, in the unsupervised setting, where no label information can be used, the diffusion may involve unreliable information and thus they may not learn reliable representations from the diffusion graph.

To address the above two problems, we propose a novel Graph Diffusion enhanced Contrastive representation Learning method (GDCL) on both topology and feature space for community detection. To tackle the first problem, different from conventional methods which only consider the topology graph and attribute information, we also consider the feature graph. We construct a nearest neighbor graph from the node attributes, which reflects the topology information of the nodes in the feature space, and apply this graph to guide the representation learning. To address the second problem, we first propose the graph diffusion module to generate a topology diffusion graph and feature diffusion graph based on topology graph and feature graph respectively. Then, we design the shared graph convolution module to learn the node representations of the local structure and global structure of the original graph and its diffusion graph. To address the unreliability problem in the diffusion graph or original graph, and learn the direct and indirect neighbor information of nodes with discriminatory power, we enhance the representation learning process by incorporating graph contrastive learning techniques. To learn a more comprehensive representation, we use the adaptive embedding fusion module together to integrate the representations in the topology graph, topology diffusion graph, attributes, feature graph, and feature diffusion graph. Finally, we provide a self-supervised strategy to train the network to obtain reliable community detection results. Our contributions are summarized as follows:

- Different from conventional methods which only consider the topology graph and feature embedding, we also consider the feature graph to improve the community detection.
- To better capture the relations and information propagation between nodes, we propose the graph diffusion module and the shared graph convolution module.
- We conduct extensive experiments on ten real datasets to demonstrate the effectiveness of the GDCL framework. Its performance outperforms existing state-of-the-art algorithms.

The subsequent structure of this paper is arranged as follows. First, in the Preliminaries and Related Work Section (see Section II), we clarify the key concepts involved in the issues explored in this paper and review previous studies closely related to the field of community detection. Moving into The Method Section (see Section III), we will elaborate in detail on the work principles of the GDCL model. Following that, in the Experiments Section (see Section IV), we comprehensively assess its performance by comparing GDCL with eight representative algorithms in the field on ten real datasets. Finally, in the Conclusion Section (see Section V), we summarize the main findings of this paper and put forward prospects for future research directions.

II. Preliminaries and Related Work

In this section, we first give the task of community detection, then present related work of community detection, graph diffusion, and graph contrastive learning.

A. Task of Community Detection

The method (GDCL) proposed in this study focuses on leveraging network structure and node attributes to identify and analyze community patterns. The key lies in employing network representation learning techniques to effectively extract the embedding representations of nodes. These embeddings are high-dimensional mathematical characterizations of nodes' positions and connection patterns within the network, capturing the intricate relationships and attributes between nodes. Once we obtain these embeddings, they can be utilized to address the problem of community detection. In addition, the symbols and terms used in the article will be clearly defined, ensuring that readers can accurately comprehend the research presented. This includes detailed explanations of core concepts such as networks, nodes, edges, and node embeddings.

The number of communities is a known quantity. The task of community detection involves grouping nodes into different communities based on their topological connections and feature similarities. Specifically, if node i is assigned to the jth community, we indicate this affiliation with $C_{ij} = 1$; conversely, if node i does not belong to the jth community, then $C_{ij} = 0$. The community affiliation of each node within the network representation is denoted by C_{ij}. This affiliation information is determined through node embeddings obtained by network representation learning. Utilizing these embedding details, and with the help of

TABLE I
NOTATIONS

Notations	Meaning
G	the undirected network
V	the collection of nodes within the graph
E	the collection of edges within the graph
$\mathbf{A}^t$	the connectivity matrix representing the topological graph
$\mathbf{A}^f$	the connectivity matrix representing the feature graph
$\mathbf{A}_{ori}$	the connectivity matrix representing the origin graph
$\mathbf{A}_{df}$	the connectivity matrix representing the diffusion graph
A^t_{ij}	node i has a link to node j within the topology graph
A^f_{ij}	node i has a link to node j within the feature graph
$\mathbf{X}$	the node features
$\hat{\mathbf{X}}$	the reconstructed node features
C	the probability matrix for assigning nodes to communities.
$\mathbf{E}_l$	the encoded representation obtained at layer l of the autoencoder.
$\mathbf{Z}^t$	node-level embeddings derived from the topological graph
$\mathbf{Z}^{t\_df}$	node-level embeddings derived from the topological diffusion graph
$\mathbf{Z}^f$	Node-level embeddings derived from the feature graph
$\mathbf{Z}^{f\_df}$	Node-level embeddings derived from the feature diffusion graph
$\mathbf{X}_{ij}$	the j-th feature of the i-th node

a community detection algorithm, we can ultimately achieve the community division for the entire network. In essence, community detection is the process of assigning nodes with similar features and tight connections to the same community, a process that relies on the node embedding information derived from network representation learning. Through this method, we are able to accurately reveal the community structure within a network. This study focuses on community detection on attribute graphs G. Let $G = \{V, E, \mathbf{X}\}$ represent an undirected network, where $V = \{v_1, \ldots, v_n\}$ represents the collection of nodes, $E = \{e_{ij}\}$ represents the collection of edges, $i, j \in \{1, \ldots, n\}$. Within the context of network representation, $\mathbf{X}$ symbolizes the ensemble of node features, and X_{ij} denotes the j-th feature of node i. The target is to create a compact, d-dimensional representation $\mathbf{z}_i \in \mathbb{R}^d$ for each node, with d being the representation's dimension, and $d(d \leq n)$. $\mathbf{A}^t$ represents the node relations of the topology graph, if A^t_{ij} is set to 1, it suggests that nodes i and j are linked by an edge within the network, and $A^t_{ij} = 0$ represents that there is no edge. $\mathbf{A}^f$ represents the relationship of the feature graph, A^f_{ij} is set to 1, it suggests that nodes i and j are linked by an edge within the network, and $A^f_{ij} = 0$ represents that there is no edge. The purpose of identifying communities within a network is to create groupings with high connectivity among members and low connectivity between different groups. Table I offers a comprehensive list of the symbols utilized in this study along with their definitions.

B. Related Work

1) Community Detection: Community detection methods can be roughly divided into two groups: topology-based community detection [20], [21], [22] and attribute community detection [23], [24], [25], [26]. Topology-based community detection methods focus on the relations between nodes in the topology space. For example, Cao et al. [27] proposes a new model for learning graph embedding, by employing a random surfing model to capture the topology of the graph directly. Wang et al. [28] proposes a model that considers both the graph structure and the community structure to obtain effective low-dimensional embeddings. Recently, Zhu et al. [29] integrates node and structure similarities into a unified similarity matrix. However, these methods ignore the node attributes, which can compensate for the limitations of the topology structure. Recent work, such as the method in [30] explores new ways of extracting multiscale information through permutation-equivariant graph framelets, which could improve community detection in heterophilous graphs. Attribute community detection methods consider both the topology and the node attributes. For example, Shchur et al. [31] utilizes Graph Convolutional Neural Networks to extract network embeddings. Wang et al. [32] observes that the relations between nodes in the feature space and in the topology space are complementary and can be adaptively fused to reveal the more intrinsic structure of communities. Similarly, Shi et al. [33] proposes a framework using framelets to learn unified semantic embeddings from multimodal data, which could inspire new approaches for integrating node attributes and graph topology in community detection. Peng et al. [34] uses an attention mechanism to consider the importance of the topology structure and node attributes of the graph to obtain higher-quality representations. Mrabah et al. [35] proposes a new graph autoencoder model to address the problems of feature randomness and feature drift. However, the above-mentioned methods fail to consider the relationships between nodes in the feature space.

2) Graph Diffusion: Graph diffusion techniques can obtain the correlations between non-adjacent neighbors by observing the propagation paths of information, thereby revealing the global structural information of the graph. Therefore, many research [36], [37], [38] have adopted graph diffusion techniques to their fields and achieved good results, such as social network analysis, node classification, and recommendation. For example, Gasteiger et al. [39] proposes an innovative message-passing method that aims to provide a more powerful, general-purpose, and spatially localized alternative to traditional graph neural networks (GNNs). The method adopts a generalized form of sparse graph diffusion that can be used for a variety of graph-based tasks. Zhao et al. [40] proposes an adaptive diffusion convolution (ADC) strategy that optimizes graph convolution by automatically learning the optimal neighborhood size in the data. Meanwhile, it does not require GNN to propagate messages through a fixed neighborhood. Some works [41], [42] replace the original graph with a diffusion graph and then combine node attribute information to learn network embedding, achieving relatively good performance. Tang et al. [36] was the first to use the diffusion process for multi-view data clustering. Tang et al. [43] proposes an innovative multi-view diffusion process that leverages higher-order contextual information to enhance pairwise affinities, which is not limited by the quality of the initial graph or the potential common subspace between multiple views. However, graph diffusion may lead to some local information loss during the diffusion process, especially when the number of iterations of random walk is limited.

3) Graph Contrastive Learning: Graph contrastive learning [44], [45], [46] is a self-supervised learning method that has been widely studied in a variety of hot research areas [44],

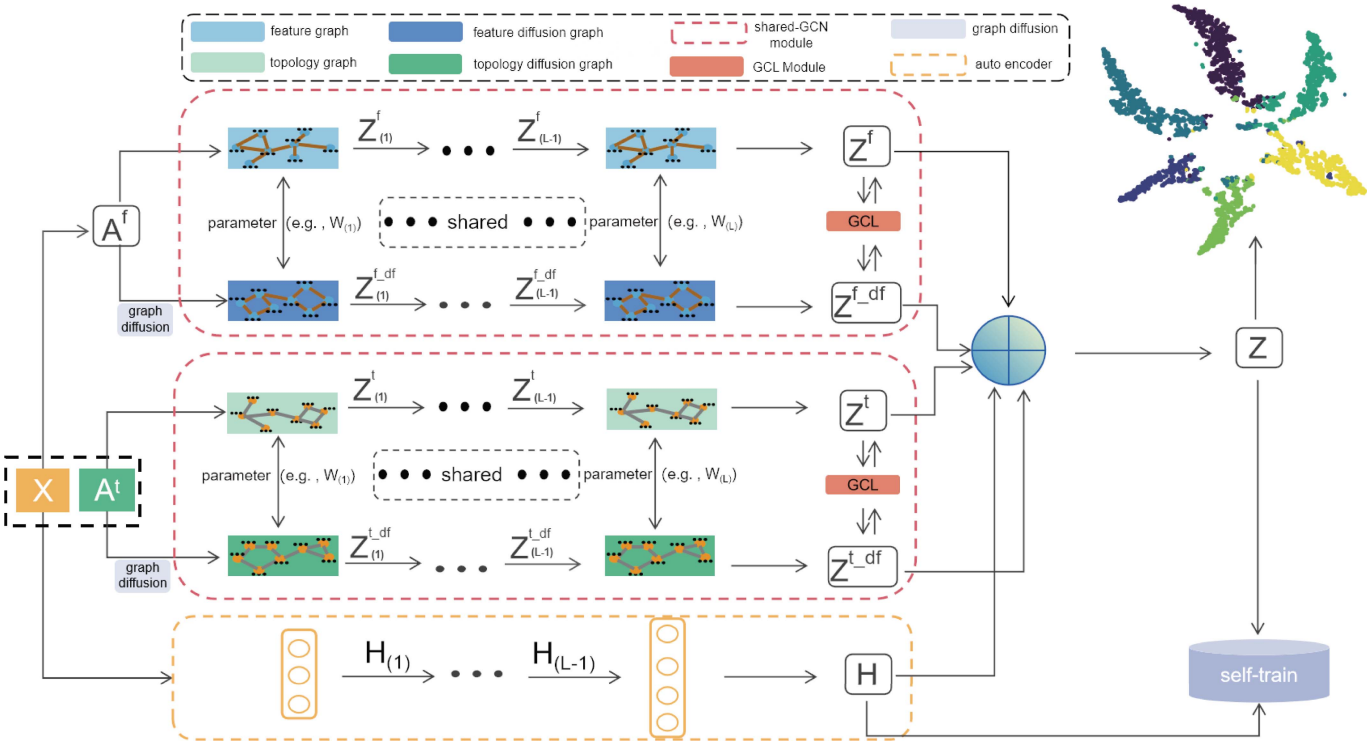


Fig. 2. The overall framework of the proposed GDCL.

[47], [48], such as node classification, recommendation, and clustering. It is effective in addressing some challenges in unsupervised learning tasks, such as the lack of real labels, false negatives, and imbalanced data distribution. Some works [49], [50] have been devoted to finding enhanced contrastive learning methods to reduce false negatives, thereby improving model performance and accuracy. Chuang et al. [51] proposes a debiased graph contrastive learning strategy to discriminate false negative samples, eliminate such bias, and thus address the issue of poor performance of hard negative mining techniques in Graph Contrastive Learning. MVGRL [18] is an algorithm that has demonstrated improved outcomes across multiple downstream applications by learning node and graph representations through the optimization of mutual information (MI) between differing views of node and graph representations, but it does not construct a feature graph to capture relationships in the feature space, nor does it apply diffusion on such a graph. SCGC [52] focuses on contrastive learning for graph clustering but primarily operates on the topology graph without integrating node attribute relationships through a feature graph. SCGDN [19] proposes the attention module (AttM) and diffusion module (DiFM) and obtains excellent embeddings. However, effectively solving the problem of false negative samples is a considerable challenge.

III. The Method

Unlike previous work, the proposed GDCL not only considers the topology graph and node attributes embeddings, but also considers the feature graph and the global structure of the graph, and integrates them into a unified framework for accurate community detection. To fully explore the information on the local and global structure of the graph, we propose a new graph contrastive learning method. Unlike conventional contrastive learning methods that obtain contrastive views by enhancing views, GDCL takes the graph and its diffusion graph as contrastive views. In the topology space, graph diffusion enables consideration of broader network connectivity, crucial for tasks like community detection where global cohesion may not be visible through local connections. In the feature space, diffusion enhances feature consistency, captures global relationships, and reduces overfitting by promoting smoothness, thus improving generalization. Fig. 2 shows the overall architecture of GDCL. The method consists of four modules: graph diffusion module, shared graph convolution module (shared-GCN), adaptive embedding fusion module, and graph contrastive learning module. In the proposed GDCL model, the key innovations are the graph diffusion enhanced contrastive learning and the shared-GCN, which synchronously learns node embeddings from both the original graph and the diffusion graph using shared parameters, enhancing the embedding quality by capturing both local and global structural information. To further improve the quality of the embedding, autoencoders and graph contrastive learning are also adopted. Specifically, in the first module, GDCL employs graph diffusion techniques to generate topology diffusion graph and feature diffusion graph. The second module utilizes a shared-weighted GCN to capture node representations of the local and global structures in the topology and feature graphs separately. The third module introduces the hidden layer adaptive embedding fusion mechanism and embedding integration strategy to obtain high-quality node representations. In the

fourth module, discriminative node representations are learned by combining the shared-GCN and the graph diffusion module.

A. Autoencoder Module

The GDCL model employs an encoder to distill a representation $\mathbf{E}$ from the input node features. In the phase dedicated to attribute encoding, we align with the approach pioneered by the AGE framework [53], which centers on the elimination of high-frequency noise from the attribute matrix $\mathbf{M}$. This process is delineated by the subsequent equation:

$$\mathbf{M} = \left(\prod_{i=1}^{l}(\mathbf{I} - \widetilde{\mathbf{L}})\right)\mathbf{X} = (\mathbf{I} - \widetilde{\mathbf{L}})^{l}\mathbf{X} \tag{1}$$

where $\mathbf{I} - \widetilde{\mathbf{L}}$ denotes the graph Laplacian as the filtering mechanism, with t representing the number of times the filtering is applied. Post-filtering, the matrix $\mathbf{M}$ is then subjected to encoding through the application of autoencoders.

We use a structure known as a Multilayer Perceptron (MLP) [54] to obtain the embedding, with the layer indexed by l can be characterized by the subsequent formulation:

$$\mathbf{E}_l = \sigma\left(\mathbf{W}_l^{\mathrm{e}}\mathbf{E}_{l-1} + \mathbf{b}_l^{\mathrm{e}}\right), \tag{2}$$

where $\mathbf{E}_l$ represents the abstracted features at at layer l, $\mathbf{W}_l^{\mathrm{e}}$ and $\mathbf{b}_l^{\mathrm{e}}$ denote the learned weights and biases at layer l, respectively. The starting point for the feature representation is signified by $\mathbf{E}_0$, which is assigned the value of the input matrix $\mathbf{M}$, and σ is utilized to introduce nonlinearity through the application of the rectified linear unit (ReLU) activation function.

In contrast, the decoder is another MLP designed to regenerate the original features from the encoded latent features, with the l-th layer described by:

$$\tilde{\mathbf{E}}_l = \sigma\left(\mathbf{W}_l^{\mathrm{d}}\tilde{\mathbf{E}}_{l-1} + \mathbf{b}_l^{\mathrm{d}}\right), \tag{3}$$

where the reconstructed feature vector at layer l is denoted by $\tilde{\mathbf{E}}_l$. The parameters within the decoder that have been adjusted through training are the weight matrix $\mathbf{W}_l^{\mathrm{d}}$ and the bias vector $\mathbf{b}_l^{\mathrm{d}}$ specific to layer l. The output from the last layer, identified as layer L, results in $\tilde{\mathbf{E}}_L$, which is utilized as the approximation $\tilde{\mathbf{X}}$ of the original dataset's features. The performance of the autoencoder is evaluated using a loss function that calculates the Frobenius norm of the discrepancy between the input features and their reconstructed counterparts:

$$\mathcal{L}_{ae} = \|\mathbf{X} - \tilde{\mathbf{X}}\|_F^2. \tag{4}$$

B. Graph Diffusion Module

Common methods for capturing global structure information of graphs include spectral methods and graph diffusion. Spectral methods, which require eigen-decomposition of graph Laplacians, can be computationally intensive for large graphs. To address this issue, GDCL adopts the graph diffusion method to capture the local and global structure of graphs. Unlike spectral methods, which rely on global eigenvalues and vectors, graph diffusion directly propagates information through the graph's structure, allowing for more efficient and scalable computation. The diffusion matrix can be used to control the probability of information diffusion on the graph. The definition of the diffusion matrix is as follows:

$$\mathcal{S} = \sum_{k=0}^{\infty}\Theta_k\mathbf{T}^k, \tag{5}$$

where $\mathbf{T} \in \mathbb{R}^{n\times n}$ is the symmetric normalized propagation matrix, and $\Theta_k \in [0, 1]$ is the weight coefficient. Specifically, we first perform symmetric normalization on the original adjacency matrix, i.e., $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, where $\mathbf{I}$ is an identity matrix. Then, we calculate the symmetric normalized propagation matrix $\widetilde{\mathbf{H}}$ as follows

$$\widetilde{\mathbf{H}} = \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}, \tag{6}$$

Finally, we iteratively calculate the final diffusion matrix $\mathbf{A}_{df}$ as follows:

$$\mathbf{A}_{df} = \xi\left(\mathbf{I}_n - (1-\xi)\widetilde{\mathbf{H}}\right)^{-1}, \tag{7}$$

where $\mathbf{I_n} \in \mathbb{R}^{n\times n}$ is the identity matrix, and ξ is a hyperparameter, typically set to 0.5.

To control the sparsity of the matrix, we adopt the steps of pruning and normalization. Specifically, we use a threshold-based pruning operation to set the smaller elements in the matrix to zero, retaining the top k largest elements. Then, we normalize it to ensure that the sum of each column of the matrix is 1.

C. Shared Graph Convolution Module

The original graph (i.e., topology graph or feature graph) and the diffusion graph are not completely independent. The original graph captures the local structure by representing immediate relationships between nodes, while the diffusion graph encapsulates the global structure by integrating higher-order connections through the diffusion process. Basically, in the graph representation learning task, the quality of the graph embedding is correlated with both the local and global structure of the graph. Therefore, we design a shared graph convolution module (shared-GCN) with shared parameters to learn the node embedding of the original graph and the diffusion graph.

First, we use shared-GCN to extract the node representation of the original graph $(\mathbf{A}^{ori}, \mathbf{X})$ as follows:

$$\mathbf{Z}_l^{ori} = ELu\left(\widetilde{\mathbf{D}}_{ori}^{-\frac{1}{2}}\widetilde{\mathbf{A}}^{ori}\widetilde{\mathbf{D}}_{ori}^{-\frac{1}{2}}\mathbf{Z}_{l-1}^{ori}\mathbf{W}_l\right), \tag{8}$$

where $\mathbf{Z}_l^{ori}$ is the embedding of the l-th layer of the original graph. $\mathbf{Z}_0 = \mathbf{X}$ represents the original node attributes, and $\mathbf{Z}_{l-1}^{ori}$ is the embedding of the $(l-1)$-th layer of the original graph. $\mathbf{W}_l$ is the network weight matrix of the l-th layer. To extract the correlated information, we design the shared-GCN, where each hidden layer shares the same weight matrix $\mathbf{W}$, and use it to learn the node representation from the diffusion graph $(\mathbf{A}^{df}, \mathbf{X})$ as follows:

$$\mathbf{Z}_l^{df} = ELu\left(\widetilde{\mathbf{D}}_{df}^{-\frac{1}{2}}\widetilde{\mathbf{A}}^{df}\widetilde{\mathbf{D}}_{df}^{-\frac{1}{2}}\mathbf{Z}_{l-1}^{df}\mathbf{W}_l\right), \tag{9}$$

where $\mathbf{Z}_l^{df}$ is the embedding of the l-th layer of the diffusion graph. By using shared parameters W_l between the original graph and the diffusion graph, the same weight matrix is used

during training to learn node representations from both the local (original graph) and global (diffusion graph) perspectives. By sharing weights, the model effectively aligns and filters the correlated information from both structures, ensuring that the learned representations achieve a harmonious blend of local and global structures.

D. Adaptive Embedding Fusion Module

GDCL uses the hidden layer adaptive embedding fusion mechanism to integrate information from each hidden layer of shared-GCN into the final layer. It then employs an embedding integration mechanism to merge node representations of local and global structures from the topology and feature graphs. Unlike [34], we naturally realize the information fusion of original graph and diffusion graph embeddings at each layer through the shared parameters in the shared-GCN, without the need for additional layer-wise fusion mechanisms. In the following, we introduce the hidden layer adaptive embedding fusion mechanism and the embedding integration strategy, respectively.

1) The Hidden Layer Adaptive Embedding Fusion Mechanism.: The existing graph representation learning methods typically only consider the last layer of node representations, which may not be able to fully capture information from different hidden layers. Different from this, we adopt the hidden layer adaptive embedding fusion mechanism to effectively utilize information from each hidden layer. Specifically, first, the embeddings of each hidden layer are concatenated, denoted as $[\mathbf{Z}_1\|\mathbf{Z}_2\|\cdots\|\mathbf{Z}_{l-1}\|\mathbf{Z}_l]$. Next, the attention coefficient $\boldsymbol{\alpha}_j$ is calculated as follows:

$$\boldsymbol{\alpha}_j = \text{Normalize}(\text{softmax}\left(\text{Tanh}([\mathbf{Z}_1\|\mathbf{Z}_2\|\cdots\|\mathbf{Z}_{l-1}\|\mathbf{Z}_l]\mathbf{W}_j)\right)), \tag{10}$$

Then, the fused node representations is calculated using $\boldsymbol{\alpha}_j$ as follows:

$$\mathbf{Z} = \boldsymbol{\alpha}_1 \cdot \mathbf{Z}_1 + \boldsymbol{\alpha}_2 \cdot \mathbf{Z}_2 + \cdots + \boldsymbol{\alpha}_{l-1} \cdot \mathbf{Z}_{l-1} + \boldsymbol{\alpha}_l \cdot \mathbf{Z}_l \tag{11}$$

We feed the fused node representations shown in (11) into the shared-GCN to obtain the last-layer embeddings $\mathbf{Z}^{ori}$ and $\mathbf{Z}^{df}$ of the original graph and diffusion graph, respectively.

2) The Embedding Integration Strategy: Using the shared-GCN and AE, we can obtain different node representations $\mathbf{Z}^t$, $\mathbf{Z}^{t\_df}$, $\mathbf{Z}^f$, $\mathbf{Z}^{f\_df}$ and $\mathbf{H}$, respectively. In consideration of the potential associations between node labels and either individual or combined embeddings, we employ an attention mechanism to learn their respective weights $[\delta_1, \delta_2, \delta_3, \delta_4, \delta_5]$. For example, for the representation $\mathbf{Z}_i$ of node i in embedding matrix $\mathbf{Z}$, we calculate its attention value δ through a non-linear transformation and a shared attention vector $p \in \mathbb{R}^{k'\times 1}$:

$$\delta_{\mathrm{i}} = \mathrm{p}^T \cdot \tanh\left(\mathrm{W} \cdot (\mathrm{Z_i})^{\mathrm{T}} + \mathrm{b}\right), \tag{12}$$

where $W \in \mathbb{R}^{k'\times k}$ is the weight matrix and k' is the number of node embeddings. Multiply these weights by their corresponding embedding and sum them up to obtain the final fused embedding Z as follows:

$$\mathbf{Z} = \boldsymbol{\delta}_1 \cdot \mathbf{Z}^t + \boldsymbol{\delta}_2 \cdot \mathbf{Z}^{t\_df} + \boldsymbol{\delta}_3 \cdot \mathbf{Z}^f + \boldsymbol{\delta}_4 \cdot \mathbf{Z}^{f\_df} + \boldsymbol{\delta}_5 \cdot \mathbf{H}, \tag{13}$$

The resulting $\mathbf{Z}$ represents the final embedding.

E. Graph Contrastive Learning Module

Based on the $\mathbf{Z}^t$, $\mathbf{Z}^{t\_df}$, $\mathbf{Z}^f$, and $\mathbf{Z}^{t\_df}$ obtained by shared-GCN, we adopt Graph Contrastive Learning Module (GCL) [44] to learn more discriminative node representations. The majority of approaches within the domain of contrastive learning typically utilize data augmentation techniques to create contrastive views of the original graph. However, due to the randomness of data augmentation, it may damage the most essential features of the graph and introduce noises. To learn the inter-information from the local structure and global structure of the network, we use the original graph and diffusion graph as contrastive views without introducing any randomness issues.

Specifically, GDCL iteratively computes the embeddings for both the local and global views of nodes within the topology graph and the feature graph. First, the process of assembling positive and negative samples adheres to a widely-used strategy. In this strategy, embeddings originating from identical nodes are categorized as positive samples, while those from distinct nodes are categorized as negative samples. To ensure a broad spectrum of negative samples, this set of nodes can be randomly selected. Second, the contrastive loss function is engineered with the goal of prompting the model to amplify the proximity of positive samples in the embedding space, while concurrently ensuring that the proximity between positive and negative samples is diminished. The similarity between two embedding vectors, Z_i and Z_j^{df}, is quantified using the cosine similarity measure, defined as follows:

$$\text{sim}\left(Z_i, Z_j^{df}\right) = \frac{Z_i^T Z_j^{df}}{\|Z_i\|\|Z_j^{df}\|}, \tag{14}$$

where Z_i and Z_j^{df} are the node representations of node i and node j in the original graph and diffusion graph, respectively. To model the positive pair similarity, we use a temperature-scaled exponential function as follows:

$$f(x) = exp\left(\frac{x}{\tau}\right), \tag{15}$$

where τ is a temperature parameter controlling the smoothness of the exponential function. The contrastive loss for a batch of positive and negative samples is formulated as:

$$\mathcal{L} = -\log\left(\frac{f(\text{sim}(Z_i, Z_i^{df}))}{\sum_{j\neq i} f(\text{sim}(Z_i, Z_j^{df})) + f(\text{sim}(Z_i, Z_i^{df}))}\right), \tag{16}$$

This loss function encourages the model to minimize the negative log-likelihood of distinguishing positive samples from a set of negative samples. According to (16), we can calculate the contrastive loss $\mathcal{L}_{cl}^t$ for the topology graph and the topology diffusion graph. Similarly, the contrastive loss for the feature graph and its associated diffusion graph is represented by $\mathcal{L}_{cl}^f$. The final contrastive loss is as follows:

$$\mathcal{L}_{cl} = \mathcal{L}_{cl}^t + \mathcal{L}_{cl}^f, \tag{17}$$

F. Objective Function

To conduct community detection, we employ the Bernoulli-Poisson algorithm (BP) [31] to calculate the loss of community detection after obtaining the final network embedding $\mathbf{Z}$. The BP model, an improved variant of the BigCLAM [8] model, is a graph generation model that can be used to detect community structures. The BP model equation is as follows:

$$A_{ij} \sim \text{Bernoulli}\left(1-\exp\left(-\boldsymbol{Z}_i\boldsymbol{Z}_j^T\right)\right), \tag{18}$$

where $\boldsymbol{Z}_i$ and $\boldsymbol{Z}_j$ are the final embedding of the i-th and j-th nodes, respectively. The negative log-likelihood of the BP model is used to calculate the loss, because directly using the probability may lead to numerical instability, whose formula is as follows:

$$-\log p(\mathbf{A}\,|\,\mathbf{Z}) = -\sum_{(i,j)\in E}\log\left(1-\exp\left(-\boldsymbol{Z}_i\boldsymbol{Z}_j^T\right)\right)+\sum_{(i,j)\notin E}\boldsymbol{Z}_i\boldsymbol{Z}_j^T, \tag{19}$$

In real-world graphs, the second term in (19) often dominates the loss because the number of non-edges is much larger than the number of edges. To address this imbalance, we use a weighted loss function that balances the two terms as follows, which is a common approach in imbalanced classification [55].

$$\begin{aligned}\mathcal{L}_{cd} = &-\mathbb{E}_{(i,j)\sim P_{(i,j)\in E}}\left[\log\left(1-\exp\left(-\boldsymbol{Z}_i\boldsymbol{Z}_j^T\right)\right)\right]\\ &+\mathbb{E}_{(i,j)\sim P_{(i,j)\notin E}}\left[\boldsymbol{Z}_i\boldsymbol{Z}_j^T\right],\end{aligned} \tag{20}$$

To improve the reliability of community detection, we propose a two-step approach that integrates the node representation $\mathbf{H}$ and the final embedding $\mathbf{Z}$. Specifically, we use the Student's t-distribution to measure the similarity between the embedding point $\mathbf{h}_i$ and the cluster center $\boldsymbol{\mu}_j$. The probability that the node i belongs to the j-th community, denoted as q_{ij} is then computed using (21).

$$q_{ij} = \frac{\left(1+\left\|\mathbf{h}_i-\boldsymbol{\mu}_j\right\|^2/v\right)^{-\frac{v+1}{2}}}{\sum_{j'}\left(1+\left\|\mathbf{h}_i-\boldsymbol{\mu}_{j'}\right\|^2/v\right)^{-\frac{v+1}{2}}}, \tag{21}$$

where v is fixed to 1in our implementation for simplicity, and P is the target distribution that computes the probability that the i-th sample belongs to the j-th cluster center, as shown in (22). This target distribution is then used to update the final embedding $\mathbf{Z}$, which can increase the compactness of the clusters by bringing samples closer to the cluster centers.

$$p_{i,j} = \frac{q_{i,j}^2/\sum_i q_{i,j}}{\sum_j' q_{i,j'}^2/\sum_i q_{i,j'}}, \tag{22}$$

Unlike traditional self-training [56], [57], which only uses the target distribution P to supervise the AE module, we propose a two-step approach that also uses P to guide the embedding $\mathbf{Z}$. This is done by adding another KL divergence term to the loss function, and an improved $\mathbf{Z}$ is expected to yield enhanced community detection outcomes as follows:

$$\begin{aligned}\mathcal{L}_{KL} &= \gamma_1 * KL(\mathbf{P}|\mathbf{Z}) + \gamma_2 * KL(\mathbf{P}|\mathbf{H}),\\ &= \gamma_1\sum_i\sum_j p_{i,j}\log\frac{p_{i,j}}{z_{i,j}} + \gamma_2\sum_i\sum_j p_{i,j}\log\frac{p_{i,j}}{q_{i,j}},\end{aligned} \tag{23}$$

where γ_1 and γ_2 serve as positive hyperparameters. Our model employs KL divergence during distribution approximation to mitigate information loss, facilitating the learning of intricate distributions. Minimizing (23) fosters a convergence toward a similar distribution for both modules. In summary, the GDCL objective function encompasses four components: community detection loss $\mathcal{L}_{cd}$, node attributes reconstruction loss $\mathcal{L}_{ae}$, KL divergence loss $\mathcal{L}_{kl}$, and GCL loss $\mathcal{L}_{cl}$. The final objective function can be expressed as:

$$\mathcal{L} = \mathcal{L}_{cd} + \mathcal{L}_{kl} + \beta 1 * \mathcal{L}_{ae} + \beta 2 * \mathcal{L}_{cl} \tag{24}$$

where $\beta 1$, $\beta 2$ are the balanced hyperparameters. where $\gamma_1, \gamma_2 > 0$ are the two hyperparameters. The training process of the proposed GDCL is shown in Algorithm 1.

Algorithm 1: Training Process of GDCL.

Input: Feature matrix $\mathbf{X}$, adjacency matrix $\mathbf{A}$, number of communities k, number of iterations I, hyperparameters β, θ, γ_1, γ_2;
Output: Node assignment result $\mathbf{C} \in \mathbb{R}^{n\times k}$;
Construct the adjacency matrix $\mathbf{A}^f$ of the feature graph from $\mathbf{X}$;
Initialize the weights and biases of the AE by pre-training AE;
Generate the topology diffusion graph $\mathbf{A}^{t\text{-}df}$ and the feature diffusion graph $\mathbf{A}^{f\text{-}df}$ using Equation 7;
Initialize the cluster centers $\boldsymbol{\mu}$ by executing the K-means clustering algorithm on the embedded feature space $\mathbf{E}$;
for $iter \leftarrow 1$ **to** I **do**
  Generate embeddings of the feature graph $\mathbf{Z}_1^f$, ... , $\mathbf{Z}_\ell^f$ using Equation 8;
  Generate embeddings of the topological graph $\mathbf{Z}_1^t$, ... , $\mathbf{Z}_\ell^t$ using Equation 8;
  Generate embeddings of the feature diffusion graph $\mathbf{Z}_1^{f\text{-}df}$, ... , $\mathbf{Z}_\ell^{f\text{-}df}$ using Equation 9;
  Generate embeddings of the topological diffusion graph $\mathbf{Z}_1^{t\text{-}df}$, ... , $\mathbf{Z}_\ell^{t\text{-}df}$ using Equation 9;
  Generate the final embeddings of the topological graph, topological diffusion graph, feature graph, and feature diffusion graph $\mathbf{Z}_\ell^t$, $\mathbf{Z}_\ell^{t\text{-}df}$, $\mathbf{Z}_\ell^f$, $\mathbf{Z}_\ell^{f\text{-}df}$ using Equation 11;
  Generate the final embedding $\mathbf{Z}$ based on the final embeddings of the topological graph, topological diffusion graph, feature graph, and feature diffusion graph using Equation 13;
  Feed the latent representation $\mathbf{E}_l$ into the autoencoder's (AE) decoding component to regenerate the node features $\mathbf{X}$. Calculate $\mathcal{L}_{cd}$, $\mathcal{L}_{ae}$, $\mathcal{L}_{cl}$, $\mathcal{L}_{kl}$ respectively;
  Compute the loss function in Equation 24 using the Adam optimizer;
end
$\mathbf{C}$ = softmax($\mathbf{Z}$);
return $\mathbf{C}$;

IV. Experiments

This section outlines the algorithms and evaluation metrics used in GDCL. It then meticulously describes the experimental

TABLE II
THE STATISTICS OF TEN DATA SETS, WHERE #NODES, #EDGES, #FEATURES, #CLASSES DENOTE THE NUMBER OF NODES, EDGES, FEATURES, AND COMMUNITIES, RESPECTIVELY

Dataset	#Nodes	#Edges	#Features	#Classes
Cornell	195	286	1,703	5
Texas	187	298	1,703	5
Washington	230	417	1,703	5
Wisconsin	265	479	1,703	5
Wiki	2,405	17,981	4,973	17
Cora	2,708	5,429	1,433	7
ACM	3,025	13,128	1,870	3
Citeseer	3,312	4,732	3,703	6
Large Cora	11,881	64,898	3,780	10
AMAC	13,752	491,722	767	10

process and the rationale behind the parameter selection. Finally, it conducts an in-depth analysis of the experimental results from multiple perspectives, including the performance assessment of the GDCL algorithm, ablation study, and visualization of the results.

A. Datasets

We evaluate the performance of GDCL on ten real-world networks with node attributes. The WebKB dataset encompasses four separate academic networks: Cornell, Texas, Washington, and Wisconsin, each delineated into five distinct clusters, and featuring a comprehensive set of 1703 attribute dimensions for nodes. The Wiki dataset represents a complex web of interconnected web pages, with nodes symbolizing individual pages and edges indicating the citation links between them. In the realm of academic collaborations, the Cora dataset stands out with its 2708 nodes, which are interconnected through 5429 edges and are categorized into seven separate communities. The Citeseer dataset, another academic network, comprises 3312 nodes linked by 4732 edges and is segmented into six communities, with nodes characterized by attribute dimensions of 3703 and 1433, respectively. The ACM dataset provides a network derived from scholarly articles, where nodes represent individual papers and edges signify co-authorship, resulting in a network with 3025 nodes and an extensive 13,128 edges across three communities. The Large Cora dataset expands on the Cora network, boasting an increased number of nodes (11,881 in total) and a significantly larger set of edges (64,898) to form 10 communities. Lastly, the AMAC dataset represents a network of co-purchased products, where nodes symbolize individual products and edges indicate frequent co-purchase relationships between them. The network consists of 13,752 nodes and 491,722 edges, with products classified into 10 distinct categories. Each node is represented by a feature vector of size 767. For a comprehensive overview of these datasets, including their specific attributes and structural details, refer to Table II.

B. Baselines and Evaluation Metrics

We compare GDCL with the following three categories of baselines that can be applied for community detection:

1) Graph embedding methods:
   - DeepWalk [20], which leverages the random walk to capture local structures.
   - LINE [21], which applies the first-order and second-order proximity to capture the similarity and structural information.
2) Attributed graph clustering methods (structure and attributes):
   - AGC [58], which employs adaptive graph convolution for global structure modeling.
   - SDCN [23], which pioneers structural information with node attributes in graph clustering.
   - FGC [59], which presents a principled approach for fine-grained attribute graph clustering.
   - VGAER [60], which utilizes variational reasoning to model both topology and attributes.
3) Graph clustering methods based on graph contrastive learning:
   - AGC-DRR [61], which designs an attribute graph clustering method to reduce information redundancy in input space and latent feature space.
   - SCGC [52], which designs a neighbor-oriented contrastive objective function, enabling the model to uphold cross-view consistency.

To assess the effectiveness of community detection, we use three well-established metrics: Accuracy (ACC), which measures the proportion of correctly assigned labels; Normalized Mutual Information (NMI), which evaluates the agreement between predicted clusters and ground truth; and Macro F1-Score (F1), which combines precision and recall. These metrics offer a comprehensive evaluation of our method's performance, ensuring robustness.

C. Experimental Setup

Within our methodology, across the entire spectrum of datasets, we use hyperparameters such as ξ, β_1, β_2, γ_1, and γ_2, respectively, and construct the feature graph using a k-nearest neighbors (k-NN) approach with k_{nn}, employing cosine similarity as the distance metric, and retaining all nearest neighbor connections without additional pruning. To ensure fair comparison, we adopt network parameter settings from [23]. The hidden layer dimensions of the Autoencoder (AE) and shared-GCN are fixed at $500 \rightarrow 500 \rightarrow 2000 \rightarrow k \rightarrow k$, where k represents the number of communities. The training procedure for GDCL consists of two steps. Firstly, the AE model is trained for 100 epochs with a fixed learning rate of 0.001. Subsequently, the entire model is trained for 500 epochs. ELu activation and batch normalization are applied to the first, second, third, and fourth layers. In the application of DeepWalk and Node2vec, the resulting embedding vectors undergo L2-norm normalization. For SDCN, the encoder's hidden layer dimensions are set to $500 \rightarrow 500 \rightarrow 2000 \rightarrow 10$, and similarly, for the Graph Convolutional Network (GCN) module, the dimensions are set to $500 \rightarrow 500 \rightarrow 2000 \rightarrow 10$ for fairness. For VGAER, we construct encoders with a 32-neuron hidden layer followed by a 16-neuron embedding layer. Finally, for AGC-DRR and SCGC, to ensure the consistency of evaluation results, we run the source

TABLE III
HYPERPARAMETERS FOR EXPERIMENTS

Model	Setting
GDCL	$\xi = 0.5$, $\beta_1 = 1000$, $\beta_2 = 0.1$, $\gamma_1 = 0.001$, $\gamma_2 = 1000$, $k_{nn} = 9$, lr = 0.001, epochs = 500
DeepWalk/Node2Vec	Window size = 10, Walk length = 80, Number of walks = 10
AGC	epochs = 50
FGC	$\alpha \in \{1e-4, 1e-2, 1, 10, 100\}$
VAGER	lr = 0.001, epochs = 1000

code of the comparative methods on all datasets. The parameters of the comparison algorithms are experimentally tuned for optimal performance. All models use the Adam optimizer. The detailed hyperparameter values can be found in Table III.

D. Result Analysis

For each dataset, we conduct a thorough evaluation by running each method 10 times and computing the average and standard deviation of these 10 best results in Table IV. In the table, the results highlighted in bold and underlined represent the best and second-best results, respectively.

Encouragingly, our method demonstrates superior community detection performance across almost all datasets, particularly excelling on the four small datasets where GDCL's performance is outstanding. Taking the Wiki dataset as an example, our proposed GDCL algorithm outperformed the runner-up approach (SCGC) by margins of 24.39%, 27.64%, and 30.48%, respectively. These enhancements underscore the efficacy of GDCL's strategy in integrating feature graph information, the global structure of the graph, and learning the local and global structure of the graph using contrastive learning.

It is noteworthy that GDCL outperforms graph embedding-based methods significantly. This superiority arises from GDCL's full exploitation of both node attributes and feature graphs, leading to a substantial enhancement in community detection performance. By leveraging this rich information, GDCL consistently achieves superior performance.

Furthermore, GDCL surpasses attribute community detection methods. This is because GDCL focuses on the feature graph, which better represents the relationships between nodes in the feature space, whereas traditional embedding methods struggle to capture these relationships effectively. In comparison with attributed graph clustering methods such as AGC, FGC, SDCN, and VGAER, which explore the topology graph and node attributes but overlook the feature graph and global structure, GDCL performs even better. In fact, the feature graph compensates for the information that the topology graph cannot capture. Not considering the global structure of the graph may lead to unstable node embeddings, while the node representations obtained through contrastive learning exhibit improved discriminability.

Contrary to methods based on the idea of contrastive learning, such as AGC-DRR and SCGC, although their performance is generally better than attribute graph clustering methods, they also do not consider the feature graph and the global structure of the graph. Therefore, the community detection performance of GDCL is still higher than them. Different from these compared methods, our approach comprehensively considers the multi-faceted information of the graph and cleverly integrates all node representations for downstream tasks.

E. Visualization

To further demonstrate the reliability of GDCL's representation, we visualize the representations of GDCL and the second-best algorithm (i.e., SCGC) on two representative datasets, Cora and Citeseer. We employ the t-Distributed Stochastic Neighbor Embedding (t-SNE [62]) algorithm to project the high-dimensional feature representations into a two-dimensional space. Subsequently, each data point is visually distinguished by associating it with a unique color corresponding to its class label. Fig. 3(a) and (b) show the visualization results of SCGC on cora and citeseer datasets, and Fig. 3(c) and (d) show the visualization results of GDCL on cora and citeseer datasets. From the figures, we can see that the visual representations obtained by GDCL are more clearly separated than those obtained by SCGC on the Cora and Citeseer datasets. Moreover, we can find that GDCL distinguishes the nodes of different communities more clearly. Furthermore, most nodes of the same color are grouped, indicating that our model can obtain high-quality embeddings.

F. Ablation Experiments

In this section, we conduct a series of ablation experiments to validate the effectiveness of our proposes method. We consider three variants of GDCL to comprehensively evaluate the performance of our method on all datasets. Specifically, we focus on the impact of feature graph, graph diffusion, and the combination of the two with contrastive learning on node embeddings:

- GDCL_wo_1: obtains node representations without considering feature graph, graph diffusion, and the combination of the two with contrastive learning.
- GDCL_wo_2: obtains node representations without considering graph diffusion and contrastive learning.
- GDCL_wo_3: obtains node representations without considering contrastive learning.

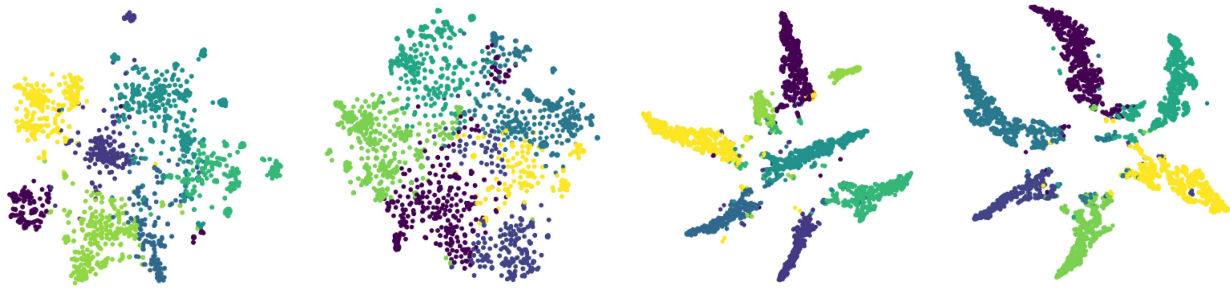
The details and pesudocodes of the three variants are provided in Section A of the supplementary materials.

Table V shows the comparison results of GDCL and its variants in terms of ACC, NMI, and F1. The results in the table is the average value of 10 independent runs. The experimental results show that GDCL_wo_3 performs worse than GDCL on all datasets, which verifies the effectiveness of the contrastive learning strategy. In addition, the results of GDCL_wo_3 are better than those of GDCL_wo_2 on most datasets, suggesting that considering both the global structure and local structure of the graph can produce higher-quality node representations. In addition, GDCL_wo_2 performs better than GDCL_wo_1 on most datasets, highlighting the effectiveness of node relations in

TABLE IV
COMMUNITY DETECTION PERFORMANCE ON TEN DATASETS

Dataset	Metrics	Deepwalk	LINE	AGC	SDCN	FGC	VGAER	AGC-DRR	SCGC	GDCL
Cornell	ACC	3.67e-1±1e-2	3.42e-1±2e-2	4.42e-1±3e-3	4.58e-1±1e-2	5.33e-1±5e-2	4.43e-1±1e-2	3.93e-1±4e-2	4.94e-1±3e-2	**5.71e-1**±2e-2
	NMI	6.54e-2±2e-2	9.80e-2±2e-2	8.20e-2±6e-2	1.22e-1±1e-2	2.66e-1±6e-2	1.31e-1±7e-3	9.00e-2±2e-2	1.80e-1±2e-2	**3.33e-1**±2e-2
	F1	2.12e-1±1e-2	2.69e-1±2e-2	1.99e-1±6e-2	2.78e-1±3e-2	3.89e-1±5e-2	3.44e-1±2e-2	2.65e-1±4e-2	3.72e-1±3e-2	**5.24e-1**±1e-2
Texas	ACC	4.29e-1±3e-2	4.69e-1±5e-2	5.50e-1±5e-2	5.86e-1±8e-3	6.63e-1±5e-2	5.33e-1±6e-3	5.30e-1±5e-3	5.49e-1±2e-2	**6.81e-1**±2e-2
	NMI	3.22e-2±1e-2	1.70e-1±1e-2	9.20e-2±1e-1	1.57e-1±3e-2	3.52e-1±9e-2	1.31e-1±9e-3	1.56e-1±8e-3	1.57e-1±2e-2	**4.03e-1**±2e-2
	F1	1.96e-1±2e-2	3.27e-1±3e-2	2.25e-1±9e-2	3.17e-1±3e-2	4.50e-1±4e-2	3.33e-1±9e-3	2.83e-1±6e-3	3.34e-1±2e-2	**4.97e-1**±2e-2
Washington	ACC	4.19e-1±3e-2	5.00e-1±3e-2	4.96e-1±4e-2	5.65e-1±2e-2	5.37e-1±2e-2	5.58e-1±1e-2	5.02e-1±4e-2	5.50e-1±9e-3	**6.78e-1**±9e-3
	NMI	4.37e-2±1e-2	1.77e-1±2e-2	8.40e-2±8e-2	1.61e-1±4e-2	2.15e-1±3e-2	1.77e-1±1e-2	1.51e-1±4e-2	1.53e-1±9e-3	**4.32e-1**±8e-3
	F1	2.40e-1±3e-2	3.24e-1±1e-2	2.03e-1±7e-2	3.03e-1±1e-2	2.85e-1±8e-3	3.46e-1±8e-3	3.31e-1±3e-2	3.47e-1±1e-2	**4.84e-1**±2e-2
Wisconsin	ACC	3.81e-1±3e-2	4.20e-1±2e-2	4.88e-1±5e-2	5.53e-1±3e-2	5.26e-1±3e-2	4.51e-1±2e-2	5.20e-1±4e-2	4.75e-1±2e-2	**5.84e-1**±3e-2
	NMI	2.67e-2±9e-3	7.90e-2±8e-3	8.30e-2±9e-2	3.32e-1±6e-2	2.45e-1±9e-2	1.12e-1±1e-2	1.23e-1±4e-2	1.50e-1±2e-2	**3.67e-1**±1e-2
	F1	2.16e-1±2e-2	2.53e-1±2e-2	1.99e-1±8e-2	4.43e-1±5e-2	2.57e-1±4e-2	3.14e-1±1e-2	3.22e-1±4e-2	3.54e-1±2e-2	**4.59e-1**±1e-2
Wiki	ACC	1.17e-1±4e-3	3.72e-1±1e-2	3.80e-1±7e-2	4.16e-1±3e-2	4.96e-1±6e-2	4.15e-1±1e-2	4.37e-1±7e-2	5.09e-1±9e-3	**5.83e-1**±5e-3
	NMI	4.33e-2±2e-3	3.52e-1±7e-3	3.62e-1±7e-2	3.61e-1±1e-2	4.90e-1±4e-2	3.64e-1±4e-3	4.15e-1±7e-2	4.62e-1±7e-3	**5.74e-1**±3e-3
	F1	8.92e-2±2e-3	3.33e-1±5e-3	2.81e-1±9e-2	2.68e-1±2e-2	4.02e-1±4e-2	3.66e-1±1e-2	3.32e-1±5e-2	4.16e-1±6e-3	**5.15e-1**±7e-3
Cora	ACC	1.81e-1±6e-3	4.36e-1±2e-2	6.10e-1±3e-2	5.70e-1±4e-2	5.84e-1±1e-1	6.75e-1±1e-2	5.97e-1±4e-2	**7.37e-1**±5e-3	7.30e-1±3e-2
	NMI	4.10e-3±1e-3	2.95e-1±1e-2	4.70e-1±3e-2	3.65e-1±3e-2	3.90e-1±1e-1	4.78e-1±7e-3	4.82e-1±3e-2	5.59e-1±6e-3	**5.63e-1**±8e-3
	F1	1.55e-1±6e-3	4.34e-1±3e-2	5.13e-1±5e-2	5.09e-1±3e-2	5.36e-1±1e-1	6.63e-1±8e-3	5.41e-1±2e-2	7.04e-1±1e-2	**7.13e-1**±4e-2
ACM	ACC	4.59e-1±2e-3	4.15e-1±2e-2	8.35e-1±0e0	7.45e-1±2e-2	8.15e-1±2e-2	5.11e-1±5e-3	8.75e-1±1e-1	8.99e-1±3e-3	**9.14e-1**±2e-3
	NMI	6.79e-2±3e-3	8.60e-2±4e-2	5.51e-1±0e0	4.42e-1±2e-2	5.19e-1±4e-2	1.86e-1±6e-3	6.31e-1±2e-1	6.67e-1±6e-3	**6.99e-1**±7e-3
	F1	3.64e-1±2e-3	3.72e-1±2e-2	8.36e-1±0e0	7.43e-1±2e-2	8.10e-1±2e-2	4.13e-1±6e-3	8.74e-1±1e-1	8.99e-1±3e-3	**9.14e-1**±2e-3
Citeseer	ACC	1.99e-1±5e-3	2.81e-1±2e-2	6.85e-1±0e0	6.20e-1±2e-2	6.12e-1±3e-2	4.75e-1±1e-2	6.77e-1±1e-2	**7.11e-1**±9e-3	6.91e-1±5e-3
	NMI	2.80e-3±8e-4	9.60e-2±8e-3	4.26e-1±0e0	3.55e-1±2e-2	3.54e-1±4e-2	2.52e-1±8e-3	4.27e-1±1e-2	**4.48e-1**±1e-2	4.36e-1±6e-3
	F1	1.77e-1±7e-3	2.48e-1±2e-2	6.38e-1±0e0	5.82e-1±1e-2	5.78e-1±3e-2	4.63e-1±1e-2	6.36e-1±1e-2	6.19e-1±1e-2	**6.53e-1**±6e-3
Large Cora	ACC	1.33e-1±3e-3	2.82e-1±2e-2	4.46e-1±3e-2	**4.71e-1**±5e-3	4.11e-1±6e-2	3.83e-1±1e-2	4.08e-1±4e-2	4.30e-1±2e-3	4.47e-1±2e-2
	NMI	1.80e-3±4e-4	1.73e-1±1e-2	3.29e-1±1e-2	1.90e-1±7e-3	2.92e-1±5e-2	2.85e-1±6e-3	2.71e-1±5e-2	2.73e-1±3e-2	**3.60e-1**±1e-2
	F1	9.64e-2±1e-3	2.21e-1±1e-2	3.14e-1±2e-2	1.85e-1±7e-3	3.33e-1±4e-2	3.27e-1±1e-2	3.21e-1±4e-2	3.14e-1±2e-2	**4.05e-1**±2e-2
AMAC	ACC	1.48e-1±8e-3	1.28e-1±3e-3	5.01e-1±4e-2	5.31e-1±2e-2	3.99e-1±6e-2	3.75e-1±4e-1	5.63e-1±3e-2	**5.82e-1**±8e-3	5.65e-1±1e-2
	NMI	1.10e-3±3e-4	8.00e-3±1e-3	3.37e-1±8e-2	3.53e-1±3e-2	2.40e-1±4e-2	7.00e-3±6e-2	4.14e-1±6e-2	3.83e-1±5e-3	**4.98e-1**±1e-2
	F1	9.62e-2±2e-3	1.01e-1±2e-3	3.48e-1±1e-1	3.10e-1±2e-2	3.16e-1±8e-2	8.80e-2±1e-1	4.23e-1±4e-2	3.89e-1±9e-3	**4.65e-1**±2e-2

Bold: the best; Underline: the second best.



(a) SCGC_Cora (b) SCGC_Citeseer (c) GDCL_Cora (d) GDCL_Citeseer

Fig. 3. Visualizations of the embeddings generated by SCGC and GDCL on the cora and citeseer datasets.

TABLE V
COMPARISON BETWEEN GDCL AND ITS VARIANTS IN TERMS OF ACC, NMI, AND F1

Metrics	Methods	Cornell	Texas	Washington	Wisconsin	Wiki	Cora	ACM	Citeseer	Large Cora	AMAC
ACC	GDCL_wo_1	4.12e-1	4.81e-1	4.89e-1	4.45e-1	4.59e-1	5.18e-1	8.36e-1	4.57e-1	3.44e-1	4.32e-1
	GDCL_wo_2	4.72e-1	5.99e-1	5.61e-1	5.13e-1	5.21e-1	6.42e-1	8.72e-1	6.22e-1	3.95e-1	4.8e-1
	GDCL_wo_3	4.77e-1	6.26e-1	5.78e-1	5.25e-1	5.43e-1	6.93e-1	8.86e-1	6.68e-1	4.22e-1	5.47e-1
	GDCL	**5.71e-1**	**6.81e-1**	**6.78e-1**	**5.84e-1**	**5.83e-1**	**7.3e-1**	**9.14e-1**	**6.91e-1**	**4.47e-1**	**5.65e-1**
NMI	GDCL_wo_1	1.82e-1	3.08e-1	2.67e-1	2.19e-1	4.15e-1	3.77e-1	5.23e-1	2.36e-1	2.47e-1	3.5e-1
	GDCL_wo_2	1.51e-1	3.19e-1	3.19e-1	3.42e-1	5.11e-1	4.79e-1	6.61e-1	3.83e-1	2.91e-1	3.86e-1
	GDCL_wo_3	2.72e-1	3.67e-1	3.06e-1	3.23e-1	5.49e-1	5.27e-1	6.52e-1	4.12e-1	3.02e-1	4.02e-1
	GDCL	**3.33e-1**	**4.03e-1**	**4.32e-1**	**3.67e-1**	**5.74e-1**	**5.63e-1**	**6.99e-1**	**4.36e-1**	**3.6e-1**	**4.98e-1**
F1	GDCL_wo_1	3.75e-1	3.89e-1	3.54e-1	3.64e-1	3.94e-1	5.05e-1	8.38e-1	4.39e-1	3.02e-1	3.42e-1
	GDCL_wo_2	4.61e-1	4.46e-1	3.74e-1	4.19e-1	4.81e-1	5.81e-1	8.92e-1	6.04e-1	3.69e-1	3.58e-1
	GDCL_wo_3	4.37e-1	4.68e-1	4.11e-1	4.49e-1	4.72e-1	6.42e-1	8.91e-1	6.31e-1	4.02e-1	4.41e-1
	GDCL	**5.24e-1**	**4.97e-1**	**4.84e-1**	**4.59e-1**	**5.15e-1**	**7.13e-1**	**9.14e-1**	**6.53e-1**	**4.05e-1**	**4.65e-1**

the feature space. It is evident that contrastive learning is highly effective for learning the local and global structures of graphs, both in the feature and topological space.

G. Sensitivity Analysis

In this section, we analyze the sensitivity of the model's performance to variations in the diffusion rate ξ, contrastive learning temperature τ, and the number of GCN layers, based on results from Tables VI, VII, and VIII. Due to space limitation, we focus on accuracy (ACC), normalized mutual information (NMI), and F1 score across three datasets: Wiki, Cora, and Citeseer.

1) Diffusion Rate ξ: The diffusion rate has a significant impact on model performance. As shown in Table VI, smaller diffusion rates, such as $\xi \leq 0.5$, typically yield good results.

2) Contrastive Learning Temperature τ: From Table VII, it can be seen that overall, the variation across the three metrics is

TABLE VI
COMMUNITY DETECTION PERFORMANCE ON DIFFERENT VALUES OF DIFFUSION RATE ξ

Metrics	Dataset	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
ACC	Wiki	**5.8e-1**	5.8e-1	5.7e-1	5.7e-1	5.7e-1	5.7e-1	5.7e-1	5.7e-1	5.7e-1	5.5e-1
	Cora	5.5e-1	**5.6e-1**	5.4e-1	5.4e-1	5.3e-1	5.3e-1	5.2e-1	5.4e-1	5.2e-1	5.0e-1
	Citeseer	6.4e-1	6.4e-1	6.5e-1	6.5e-1	**6.5e-1**	6.4e-1	6.5e-1	6.4e-1	6.5e-1	6.4e-1
NMI	Wiki	5.7e-1	5.7e-1	5.6e-1	5.7e-1	5.7e-1	5.7e-1	5.6e-1	**5.7e-1**	5.7e-1	5.6e-1
	Cora	**3.5e-1**	3.4e-1	3.4e-1	3.4e-1	3.4e-1	3.3e-1	3.4e-1	3.4e-1	3.3e-1	3.0e-1
	Citeseer	3.7e-1	3.8e-1	3.8e-1	3.8e-1	3.8e-1	3.8e-1	**3.8e-1**	3.8e-1	3.7e-1	3.7e-1
F1	Wiki	5.1e-1	5.1e-1	**5.1e-1**	5.0e-1	5.0e-1	5.0e-1	5.0e-1	5.1e-1	5.1e-1	4.7e-1
	Cora	5.2e-1	**5.4e-1**	5.2e-1	5.2e-1	5.1e-1	5.1e-1	4.9e-1	5.1e-1	4.9e-1	4.8e-1
	Citeseer	6.1e-1	6.1e-1	6.1e-1	6.1e-1	6.1e-1	6.1e-1	**6.1e-1**	6.1e-1	6.1e-1	6.1e-1

TABLE VII
COMMUNITY DETECTION PERFORMANCE ON DIFFERENT VALUES OF CONTRASTIVE LEARNING TEMPERATURE τ

Metrics	Dataset	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
ACC	Wiki	5.7e-1	5.8e-1	**5.9e-1**	5.8e-1	5.8e-1	5.8e-1	5.8e-1	5.8e-1	5.7e-1	5.8e-1
	Cora	5.3e-1	5.2e-1	5.2e-1	5.3e-1	5.4e-1	5.5e-1	5.4e-1	5.3e-1	5.4e-1	**5.5e-1**
	Citeseer	6.4e-1	**6.5e-1**	6.4e-1	6.4e-1	6.4e-1	6.5e-1	6.4e-1	6.4e-1	6.4e-1	6.4e-1
NMI	Wiki	5.7e-1	5.7e-1	**5.7e-1**	**5.7e-1**	5.7e-1	5.7e-1	5.7e-1	5.7e-1	5.7e-1	5.7e-1
	Cora	3.4e-1	3.4e-1	3.4e-1	**3.4e-1**	3.4e-1	3.4e-1	3.4e-1	3.4e-1	**3.4e-1**	**3.4e-1**
	Citeseer	3.7e-1	3.8e-1	3.8e-1	3.7e-1	3.8e-1	**3.8e-1**	3.8e-1	**3.8e-1**	3.8e-1	3.8e-1
F1	Wiki	5.1e-1	5.1e-1	5.2e-1	5.1e-1	5.1e-1	5.1e-1	5.1e-1	5.1e-1	5.1e-1	**5.2e-1**
	Cora	4.9e-1	5.0e-1	4.9e-1	5.0e-1	5.1e-1	**5.2e-1**	5.1e-1	5.0e-1	5.2e-1	**5.2e-1**
	Citeseer	6.0e-1	6.1e-1	6.1e-1	6.1e-1	6.1e-1	**6.2e-1**	6.1e-1	6.1e-1	6.1e-1	6.1e-1

TABLE VIII
COMMUNITY DETECTION PERFORMANCE ON DIFFERENT VALUES OF NUMBER OF GCN LAYERS

Metrics	Dataset	layer1	layer2	layer3
ACC	Wiki	5.0e-1	5.1e-1	**5.3e-1**
	Cora	5.9e-1	**6.6e-1**	6.3e-1
	Citeseer	5.9e-1	**6.3e-1**	6.1e-1
NMI	Wiki	5.1e-1	5.2e-1	**5.4e-1**
	Cora	4.8e-1	**5.0e-1**	4.9e-1
	Citeseer	3.5e-1	**3.7e-1**	**3.7e-1**
F1	Wiki	4.3e-1	4.2e-1	**4.7e-1**
	Cora	5.6e-1	5.7e-1	**6.0e-1**
	Citeseer	5.7e-1	5.4e-1	**5.8e-1**

not large, indicating that the contrastive learning temperature τ does not have a substantial effect on the results, and the model is not highly sensitive to this parameter.

3) Number of GCN Layers: From the F1 score metric, it can be seen that the model performs better when the number of GCN layers is 3, and for the Wiki dataset, the performance across all metrics reaches its optimal point when the GCN layer count is 3. Overall, the results from Table VIII suggest that increasing the number of layers helps improve model performance across all metrics.

In conclusion, the sensitivity analysis indicates that a smaller diffusion rate ξ, along with deeper GCN layers, leads to the best overall model performance. These findings can guide the fine-tuning of the model's hyperparameters to achieve optimal results.

V. CONCLUSION

In this paper, we propose a novel Graph Diffusion enhanced Contrastive representation Learning method named GDCL on topology and feature space for community detection. Our method successfully addresses two challenges that existed in previous works: the insufficient consideration of the relationships between nodes in the feature space and the inadequate exploration of the global structure of the graph. To tackle the first challenge, GDCL constructs a feature graph to uncover relationships among nodes in the feature space. For the second challenge, we introduce a graph diffusion module to capture the global structures of the original graph (i.e., topology graph or feature graph). To learn high-quality representations of these graphs, we propose the shared-GCN and GCL. Additionally, we utilize an adaptive embedding fusion module to obtain enhanced node representations for accurate community detection. Comparative experiments on ten real-world datasets, spanning various types of networks, demonstrate that GDCL outperforms the state-of-the-art methods, highlighting its efficiency and practicality.

Nonetheless, further exploration is needed to understand how GDCL performs across different network types and to identify any performance differences that may arise. Moreover, while our experiments were conducted on networks of moderate size, investigating the scalability of GDCL to large-scale networks is crucial for real-world applications. Therefore, in future work, we aim to explore the applicability and performance differences of GDCL across diverse network structures and to extend our model to efficiently handle large-scale networks. We also plan to apply our model to other downstream tasks such as node classification and link prediction.

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. 29th 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, 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] R. Guimera, S. Mossa, A. Turtschi, and L. N. Amaral, "The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles," *Proc. Nat. Acad. Sci.*, vol. 102, no. 22, pp. 7794–7799, 2005.

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

[8] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Proc. 6th ACM Int. Conf. Web Search Data Mining*, Rome, Italy, 2013, pp. 587–596.

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

[10] F. Sun, M. Qu, J. Hoffmann, C. Huang, and J. Tang, "vGraph: A generative model for joint community detection and node representation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 512–522.

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

[12] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity based community detection with deep learning," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, New York, NY, USA, 2016, pp. 2252–2258.

[13] F. Ye, C. Chen, and Z. Zheng, "Deep autoencoder-like nonnegative matrix factorization for community detection," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Torino, Italy, 2018, pp. 1393–1402.

[14] 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.

[15] D. He et al., "Community-centric graph convolutional network for unsupervised community detection," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2021, pp. 3515–3521.

[16] M. Li et al., "Guest editorial: Deep neural networks for graphs: Theory, models, algorithms, and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 4367–4372, Apr. 2024.

[17] E. Buchnik and E. Cohen, "Bootstrapped graph diffusions: Exposing the power of nonlinearity," in *Proc. Abstr. ACM Int. Conf. Meas. Model. Comput. Syst.*, 2018, pp. 8–10.

[18] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 4116–4126.

[19] Y. Ma and K. Zhan, "Self-contrastive graph diffusion network," in *Proc. 31st ACM Int. Conf. Multimedia*, 2023, pp. 3857–3865.

[20] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.

[21] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.

[22] A. Grover and J. Leskovec, "node2Vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, San Francisco, CA, USA, 2016, pp. 855–864.

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

[24] 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.

[25] A. Tsitsulin, J. Palowitch, B. Perozzi, and E. Müller, "Graph clustering with graph neural networks," *J. Mach. Learn. Res.*, vol. 24, no. 127, pp. 1–21, 2023.

[26] L. Zhang, Z. Wu, H. Yang, W. Zhang, and P. Zhou, "Feature graph augmented network representation for community detection," *IEEE Trans. Comput. Social Syst.*, vol. 11, no. 6, pp. 7516–7527, Dec. 2024.

[27] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 1145–1152.

[28] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 203–209.

[29] 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.

[30] J. Li, R. Zheng, H. Feng, M. Li, and X. Zhuang, "Permutation equivariant graph framelets for heterophilous graph learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 9, pp. 11634–11648, Sep. 2024.

[31] O. Shchur and S. Günnemann, "Overlapping community detection with graph neural networks," *Deep Learn. Graphs Workshop*, 2019.

[32] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei, "AM-GCN: Adaptive multi-channel graph convolutional networks," in *Proc. 26th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, Virtual Event, CA, USA, 2020, pp. 1243–1253.

[33] J. Shi, M. Li, Y. Chen, L. Cui, and L. Bai, "Multimodal graph learning with framelet-based stochastic configuration networks for emotion recognition in conversation," *Inf. Sci.*, vol. 686, 2025, Art. no. 121393.

[34] Z. Peng, H. Liu, Y. Jia, and J. Hou, "Attention-driven graph clustering network," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 935–943.

[35] 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.

[36] C. Tang et al., "CGD: Multi-view clustering via cross-view graph diffusion," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 5924–5931.

[37] B. Jiang, D. Lin, J. Tang, and B. Luo, "Data representation and learning with graph diffusion-embedding networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10406–10415.

[38] L. Zhang, Y. Liu, X. Zhou, C. Miao, G. Wang, and H. Tang, "Diffusion-based graph contrastive learning for recommendation with implicit feedback," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2022, pp. 232–247.

[39] J. Gasteiger, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 13366–13378.

[40] J. Zhao, Y. Dong, M. Ding, E. Kharlamov, and J. Tang, "Adaptive diffusion in graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 23321–23333.

[41] J. Klicpera, A. Bojchevski, and S. Günnemann, "Propagate: Graph Neural Networks meet Personalized PageRank," in *Proc. Int. Conf. Learn. Representations*, New Orleans, LA, USA, May 2019.

[42] A. Tsitsulin, D. Mottin, P. Karras, and E. Müller, "VERSE: Versatile graph embeddings from similarity measures," in *Proc. 2018 World Wide Web Conf.*, 2018, pp. 539–548.

[43] Q. Li, S. An, L. Li, W. Liu, and Y. Shao, "Multi-view diffusion process for spectral clustering and image retrieval," *IEEE Trans. Image Process.*, vol. 32, pp. 4610–4620, 2023.

[44] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 5812–5823.

[45] D. Xu, W. Cheng, D. Luo, H. Chen, and X. Zhang, "InfoGCL: Information-aware graph contrastive learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 30414–30425.

[46] J. Xia, L. Wu, J. Chen, B. Hu, and S. Z. Li, "SimGRACE: A simple framework for graph contrastive learning without data augmentation," in *Proc. ACM Web Conf.* 2022, pp. 1070–1079.

[47] Y. Zhang et al., "Enhancing sequential recommendation with graph contrastive learning," in *Proc. Thirty-First Int. Joint Conf. Artif. Intell.*, Vienna, Austria, 2022.

[48] Y. Li, P. Hu, Z. Liu, D. Peng, J. T. Zhou, and X. Peng, "Contrastive clustering," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 8547–8555.

[49] J. Xia, L. Wu, G. Wang, J. Chen, and S. Z. Li, "ProGCL: Rethinking hard negative mining in graph contrastive learning," in *Proc. Int. Conf. Mach. Learn.*, Baltimore, Maryland, 2022, pp. 24332–24346.

[50] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proc. Web Conf.*, 2021, pp. 2069–2080.

[51] C. -Y. Chuang, J. Robinson, Y. -C. Lin, A. Torralba, and S. Jegelka, "Debiased contrastive learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 8765–8775.

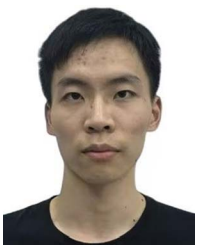
[52] Y. Liu et al., "Simple contrastive graph clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 10, pp. 13789–13800, Oct. 2024.

[53] G. Cui, J. Zhou, C. Yang, and Z. Liu, "Adaptive graph encoder for attributed graph embedding," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 976–985.

[54] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—A review of applications in the atmospheric sciences," *Atmospheric Environ.*, vol. 32, no. 14/15, pp. 2627–2636, 1998.

[55] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: A review," *GESTS Int. Trans. Comput. Sci. Eng.*, vol. 30, no. 1, pp. 25–36, 2006,.
[56] J. Xie, R. B. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 478–487.
[57] H. Sun et al., "Network embedding for community detection in attributed networks," *ACM Trans. Knowl. Discov. Data*, vol. 14, no. 3, pp. 1–25, 2020.
[58] X. Zhang, H. Liu, Q. Li, and X. Wu, "Attributed graph clustering via adaptive graph convolution," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 4327–4333.
[59] Z. Kang, Z. Liu, S. Pan, and L. Tian, "Fine-grained attributed graph clustering," in *Proc. 2022 SIAM Int. Conf. Data Mining*, Alexandria, VA, USA, 2022, pp. 370–378.
[60] C. Qiu, Z. Huang, W. Xu, and H. Li, "VGAER: Graph neural network reconstruction based community detection," 2022, *arXiv:2201.04066*.
[61] L. Gong, S. Zhou, W. Tu, and X. Liu, "Attributed graph clustering with dual redundancy reduction," in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 3015–3021.
[62] L. van der Maaten and G. E. Hinton, "Visualizing data using T-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.

Lei Zhang (Senior Member, IEEE) received the B.Sc. degree from Anhui Agriculture University, Hefei, China, in 2007, and the Ph.D. degree from the University of Science and Technology of China, Hefei, in 2014. He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University, Hefei. He has authored or coauthored more than 100 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. His main research interests include multi-objective optimization and their applications, data mining, machine learning, social network analysis and recommendation. He was the recipient of the ACM CIKM'12 Best Student Paper Award. He is a member of ACM.



Fubo Sun received the B.Sc. degree from the School of Information Management and Information System, China Pharmaceutical University, Nanjing, China, in 2024. He is currently working toward the master's degree with the School of Computer Science and Technology, Anhui University, Hefei, China. His current research interests include graph embedding and community detection.

Zeqi Wu received the B.Sc. degree from the School of Computer Science and Technology, Jinggangshan University, Ji'an, 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 current research interests include graph embedding and community detection.

Wuji Zhang received the B.Sc. degree from the School of Computer Science and Technology, Anhui agricultural University of Technology, Hefei, China, in 2021. He is currently working toward the master's degree from the School of Computer Science and Technology, Anhui University, Hefei. His current research interests include graph neural network, social recommendation system, and multi-behavior recommendation system.

Haipeng Yang (Member, IEEE) received the B.Sc. degree from the School of Computer Science and Technology, Anhui University, Hefei, China, in 2019. He is currently working toward the Ph.D. degree with the School of Computer Science and Technology, Anhui University. His research interests include multi-objective optimization and social network analysis.

Peng Zhou (Senior Member, IEEE) received the B.Sc. degree in computer science and technology from the University of Science and Technology of China, Hefei, 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. He has authored or coauthored 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. His research interests include machine learning, data mining, and artificial intelligence.