Deep Masked Graph Node Clustering

Jinbin Yang, Jinyu Cai, Luying Zhong, Yueyang Pi, and Shiping Wang, *Senior Member, IEEE*

***Abstract*—In recent years, reconstructing features and learning node representations by graph autoencoders (GAE) have attracted much attention in deep graph node clustering. However, existing works often overemphasize structural information and overlook the impact of real-world prevalent noise on feature learning and clustering with graph data, which may be detrimental to robust training. To address these issues, the utilization of a masking strategy that specifically focuses on feature reconstruction may mitigate these limitations. In this article, we propose a graph node clustering generative method named deep masked graph node clustering (DMGNC), which leverages a masked autoencoder to effectively reconstruct node features, enabling the discovery of latent information crucial for accurate node clustering. Additionally, a clustering self-optimization module is designed to guide the iterative update of our end-to-end clustering framework. Further, we extend the masked graph autoencoder (MGA) and develop a contrastive method called deep masked graph node contrastive clustering (DMGNCC), which applies the MGA to graph node contrastive learning at both the node level and the class level in a united model. Extensive experimental results on real-world graph benchmark datasets demonstrate the effectiveness and superiority of the proposed method.**

***Index Terms*—Deep clustering, deep learning, graph node clustering, neural networks, unsupervised learning.**

I. Introduction

GRAPH data have become increasingly widespread and significant in society today. They are involved in various fields such as machine learning [1] and data mining [2]. In these fields, graph node clustering is a fundamental problem in graph analysis and has numerous applications in recommender systems [3] and social network [4]. The goal of graph node clustering is to assign similar nodes to the same class based on edge weights or edge distances, resulting in nodes within the same class as similar as possible and nodes in different classes being as dissimilar as possible. Due to the complexity and irregularity of graph data, traditional clustering algorithms are often incapable of dealing effectively in this field. As a result, graph node clustering has become a popular research topic.

Manuscript received 17 May 2023; revised 7 April 2024; accepted 30 April 2024. This work was supported in part by the National Natural Science Foundation of China under Grant U21A20472 and Grant 62276065, and in part by the National Key Research and Development Plan of China under Grant 2021YFB3600503. *(Jinbin Yang and Jinyu Cai contributed equally to this work.) (Corresponding author: Shiping Wang.)*

The authors are with the College of Computer and Data Science and Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fuzhou 350116, China (e-mail: yangjinbinfzu@163.com; Jinyucai1995@gmail.com; luyingzhongfzu@163.com; piyueyangcc@163.com; shipingwangphd@163.com).

Digital Object Identifier 10.1109/TCSS.2024.3401218

The primary challenge of graph node clustering lies in the representation of nodes. Nodes typically contain both attribute features and relationship information, making it challenging to find suitable representations. Traditional clustering algorithms designed for Euclidean data, such as k-means-based [5], [6], [7], [8], [9] and spectral clustering-based approaches [10], [11], [12], are not directly applicable to graph data due to the complex structure and connectivity of graphs. They ignore the structure of the graph or the characteristics of the nodes, limiting their ability to effectively utilize the interaction between node content information and structural information in graph data. Graph theory and game theory [13], as well as the belief dynamics model [14], [15], can effectively achieve graph clustering. However, compared to traditional clustering problems, graph node clustering requires consideration of node similarity and relative positions, necessitating the use of specific graph embedding techniques to map nodes to a low-dimensional vector space for clustering. Recently, deep learning methods [12], [16], [17], [18], [19], [20], [21], [22] have shown superior performance against traditional methods in graph clustering tasks, and graph neural networks [23], [24], [25], [26], [27] have emerged as a popular approach for learning node representations. Graph neural networks capture both local and global structural properties of the graph and can be trained both using supervised and unsupervised learning methods to predict node labels or cluster assignments. The most widely used model for graph neural networks is the graph autoencoder (GAE) [28], [29], which compresses and reconstructs input data by encoding them into a low-dimensional representation and then decoding them back into the original space to learn the node representations. However, traditional GAEs often encounter the issue of oversmoothing [30], where the learned node representations become too similar, leading to a loss of discrimination between nodes. Measuring the similarity between nodes is another challenge in graph node clustering. Unlike traditional clustering problems where Euclidean distance or cosine similarity are commonly utilized, graph node clustering requires a specific similarity measure to capture the nonlinear relationships between nodes.

Currently, deep graph node clustering methods [31] can be broadly classified into two categories: generative and contrastive methods. Generative graph clustering methods typically apply GAEs to reconstruct the structural information or node features of the graph. Recently, the trend has been to reconstruct the adjacency matrix of the graph data to further explore the structural information of the graph. For example, Pan et al. [32] proposed an adversarial graph embedding framework for graph data that trained decoders to reconstruct the graph structure. Similarly, Wang et al. [33] combined an attention mechanism

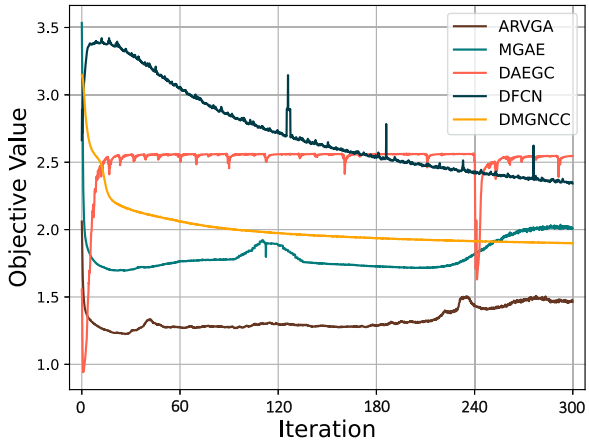

Fig. 1. Convergence curves of five graph node clustering methods on the ACM dataset.

with an autoencoder to reconstruct the adjacency matrix by self-optimization. Tu et al. [34] developed a symmetric GAE to further reconstruct the adjacency matrix using the latent feature representation reconstructed by the graph decoder. However, these methods may overemphasize the structural information of the graph, while ignoring the importance of node features.

In contrastive methods, the goal is to learn representations that can distinguish between positive and negative samples. Positive samples refer to pairs of nodes belonging to the same cluster, while negative samples are pairs of nodes belonging to different clusters. The contrastive loss function encourages the model to map positive samples close together in the representation space while pushing negative samples farther apart. To improve the quality of learned representations, contrastive methods often use various data augmentation techniques such as masked attributes, subgraph sampling, and randomly added or removed edges for data augmentation. Zhu et al. [35] designed an augmentation scheme based on the node centrality measures and added more noise to unimportant node features to achieve adaptive augmentation contrast. Zhao et al. [36] developed a unified graph debiased contrastive learning framework that jointly performed graph representation learning and clustering tasks.

Although optimal performance has been achieved in prior research, however, we found that existing works overemphasize structural information and the feature reconstruction without masking may not be conducive to robust training. We illustrate this phenomenon on the ACM dataset in Fig. 1. The previously mentioned methods, ARVGA [32], DAEGC [33], and DFCN [34], for reconstructing the adjacency matrix exhibit fluctuating objective values and unstable convergence during training, which could lead to poor robust training of the learned node representations. In addition, marginalized graph autoencoder (MGAE) proposed by Wang et al. [37] utilized marginalized graph convolutional networks (GCNs) to corrupt the network node content though, as well as features of marginalized corruption. However, the same instability issue persists when utilizing GAEs. Recently, using masking strategies to reconstruct features, so that the model can learn and provide more potential information for unsupervised tasks has great potential and has been widely verified and applied in computer vision [38], [39] and natural language processing [40]. A self-supervised masked graph autoencoder (MGA) [41] experimentally found that simply reconstructing node features can enable the model to learn sufficient valid information, thereby providing better assistance for downstream tasks. However, the above methods have mainly been proposed for tasks such as node classification and transfer learning, while the application of masked feature reconstruction in deep graph node clustering remains unexplored. Based on these, we propose a generative method named deep masked graph node clustering (DMGNC), which is capable of robust reconstruction and effective clustering to address the two challenges pointed out above. As shown in Fig. 2, the framework of DMGNC mainly consists of a masked graph representation learning module and a self-optimizing clustering module. In the masked graph representation learning module, a portion of nodes are randomly selected, and their features are replaced with masks to learn latent representations by encoder and reconstruct features by decoder. In the self-optimizing clustering module, the encoder-learned and remasked latent representations are unified in an end-to-end unified framework and use the self-training module to guide the optimization process with soft labels for better training process feedback. In addition, we propose a contrastive method called deep masked graph node contrastive clustering (DMGNCC) based on the MGA of DMGNC, which applies the MGA to compare the similarity of positive and negative samples, respectively. As shown in Fig. 3, the framework of DMGNCC is composed of a pair-masked network (PMN), a node-level contrastive module (NCM), and a class-level contrastive module (CCM). Two data-augmented graph views are constructed by PMN, and then node-level and class-level contrastive learning are performed in an end-to-end model, respectively.

The main contributions can be summarized as follows.

1) Propose a DMGNC that uses the masked latent representation and performs feature reconstruction in an end-to-end unified framework to learn graph embeddings combined with auxiliary target distributions and clustering results.
2) Present an extended version, proposing a DMGNCC, using MGA for feature reconstruction of positive and negative samples and graph node contrastive learning from node level and class level.
3) The proposed methods integrate structural and informational features more efficiently and enable more effective and robust training under the prevalent noise in the real-world scenario.
4) Extensive experiments on six datasets demonstrate that our methods are competitive and significantly outperform graph node clustering baselines.

The rest of this article is arranged as follows. In Section II, we review related research on generative graph node clustering and contrastive graph node clustering. In Section III, we present the proposed DMGNC and DMGNCC methods. In Section IV, extensive experiments utilizing real-world datasets are conducted

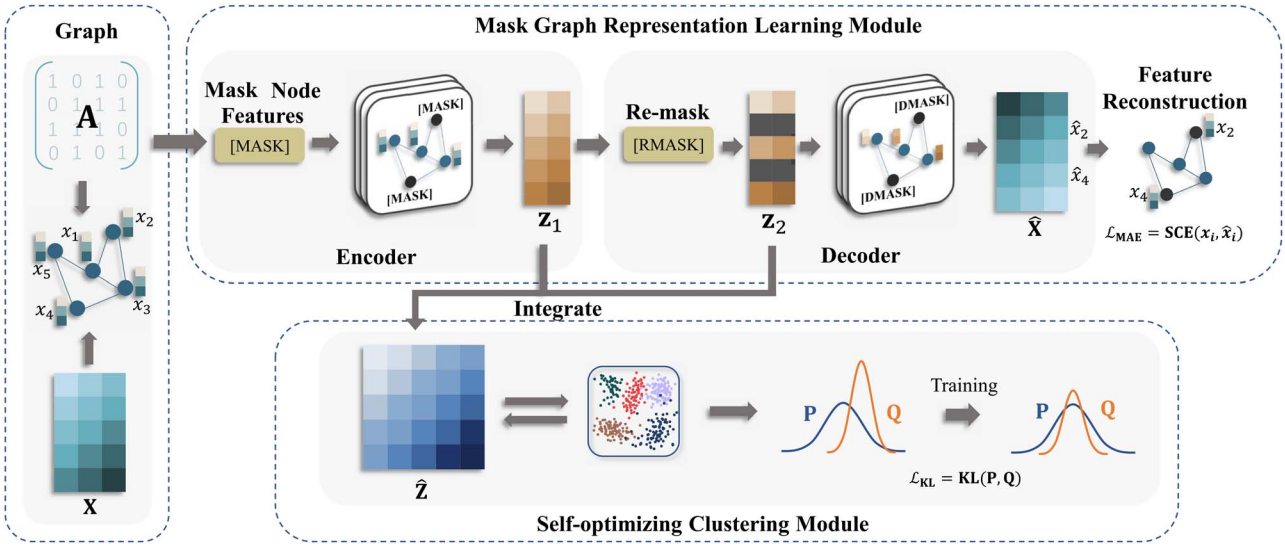

Fig. 2. Architecture of the proposed DMGNC, which is a joint framework for mask feature reconstruction and performing clustering tasks. The framework involves two essential parts: mask graph representation learning module and self-optimizing clustering module. The mask graph representation learning module performs masking, encoding, remasking, and decoding operations by randomly selecting nodes, respectively. The encoder-learned and remasked latent representations are then unified and optimized in the self-optimizing clustering module.

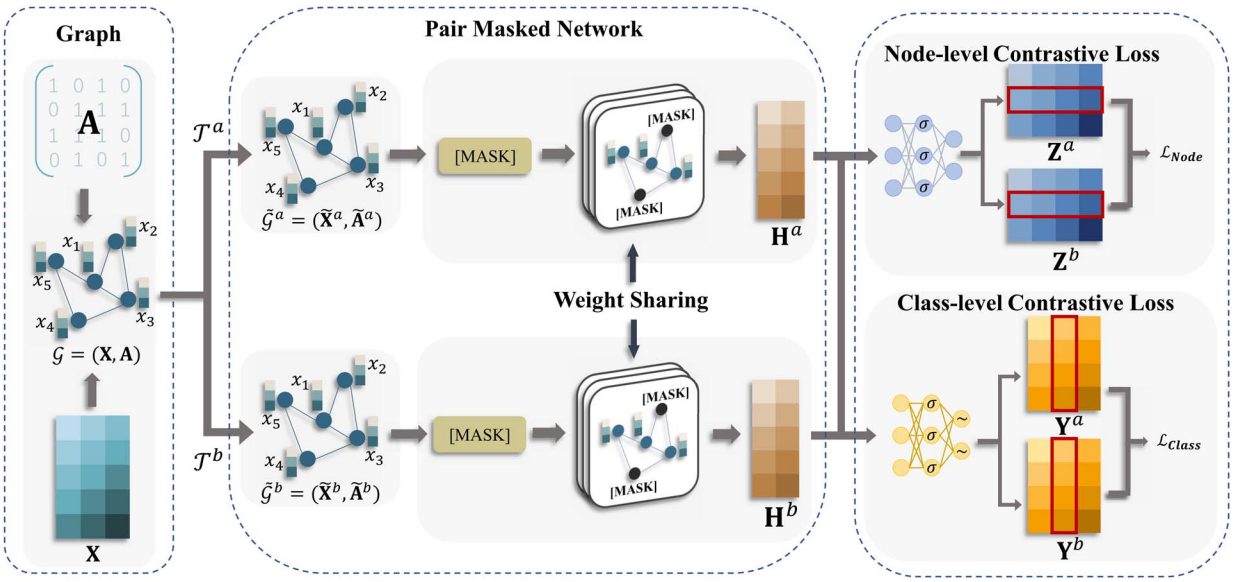

Fig. 3. Architecture of the proposed DMGNCC, which is a unified framework for contrastive clustering learning in masked networks using two data-augmented views. The framework involves three essential parts: PMN, NCM, and CCM. Mask feature operations are performed in a parameter-sharing pairwise mask network to learn the node embedding representations of the two augmented views, which are then optimized uniformly from node-level contrastive loss to class-level contrastive loss, respectively.

to validate the clustering performance of the proposed methods. In Section V, this article is concluded.

II. Related Work

A. Generative Graph Node Clustering

Generative methods learn representations of graph nodes mainly by designing loss functions in the output space. GAE [42] is a widely used generative method. The main idea of GAE is to employ a GCN [43] as an encoder to embed the graph into a low-dimensional vector space while preserving the graph structure and node attributes, and finally to explore potential representations by reconstructing the adjacency matrix. Subsequently, many effective applications have been proposed based on GAE. Wang et al. [33] proposed deep attention-embedded graph clustering that combined attention mechanisms into graph clustering and employed autoencoders to learn latent representations. Park et al. [44] presented a symmetric graph convolutional autoencoder with Laplacian smoothing for the encoder and Laplacian sharpening for the decoder,

which generated a low-dimensional latent representation from a graph. Wang et al. [37] added marginalization mechanisms to the GAE, dynamically adjusting the topology and content information by adding random noise. Although these methods have achieved better clustering results, there is still an over-smoothing phenomenon similar to GAE, where the learned node representations tend to be similar, resulting in a negative impact on the clustering performance. Bo et al. [45] have since proposed to jointly learn GAEs within a unified framework to alleviate the over-smoothing problem through information transfer operations. He et al. [46] developed an adaptive graph convolutional clustering model to update the graph structure and data representation layers. However, these works overemphasize the structural information of nodes, and robust training may be better served by using masked feature reconstruction.

B. Contrastive Graph Node Clustering

Due to the remarkable success of contrastive learning in the field of computer vision [47], [48], its application to graph node clustering has gradually gained favor. Contrastive learning primarily employs data augmentation to obtain different relevant views of the given data and constructs objective functions that maximize the consistency between positive and negative samples in the latent space. Currently, a variety of graph augmentation techniques, including node dropping, edge modification, and subgraph construction, have been introduced in contrastive learning for graph data. Additionally, previous work [49] has indicated that there is no single data augmentation technique for graphs that consistently outperforms others. Velicković et al. [50] was inspired by Deep InfoMax [51] and aimed to learn node representations by maximizing the mutual information between local patches of a graph. Similarly, Hassani and Ahmadi [52] used InfoMax loss to maximize the mutual information between different views for learning node representations. Zhu et al. [53] augmented views by randomly perturbing nodes or edges and their features and learns node representations by maximizing the consistency between two views. Xia et al. [54] bridged augmented views by operations such as pulling positive samples closer and pushing negative samples further. Liu et al. [55] exploited the data itself to generate an optimal graph structure and used contrastive learning to maximize the agreement between the learned topology and the self-augmented learning objective. Devvrit et al. [56] proposed a method for scalable graph clustering with side information on node features, capable of utilizing contrastive learning with graph neural networks and node features to learn clusterable features. Although these methods have achieved promising results, there is a lack of research that combines MGA with contrastive learning, which could potentially lead to even better performance in graph node clustering.

III. Proposed Method

First, we present the frequently used mathematical notations. An undirected graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ with N number of nodes, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ and $\mathcal{E} = \{e_{ij}\}$ are a set of nodes and a set of edges between nodes, respectively.

TABLE I
Glossary of Notations in This Article

Notations	Descriptions
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$	A graph $\mathcal{G}$ consisting of a set of nodes $\mathcal{V}$, a set of edges $\mathcal{E}$ and a feature matrix $\mathbf{X}$
v_i	A node $v_i \in \mathcal{V}$
e_{ij}	An edge $e_{ij} \in \mathcal{E}$
k	Number of clusters
N	The number of nodes with $N = \lvert\mathcal{V}\rvert$
d	The dimension of node feature vector
$\mathbf{x}_i$	The d-dimensional feature vector for node v_i
$\mathbf{A}$	The graph adjacency matrix
$f_{\boldsymbol{\theta}_e}$	Encoder
$f_{\boldsymbol{\theta}_d}$	Decoder
$\mathbb{R}$	Euclidean space
$\mathbf{Z}$	Latent representation
$\tilde{\mathbf{X}}$	Masked feature matrix
$\tilde{\mathbf{X}}'$	Reconstruction feature matrix
$\mathbf{Q}$	Clustering result distribution
$\mathbf{P}$	Target distribution
γ	A hyperparameter of the scaling factor
$\lVert \cdot \rVert$	The ℓ_2-normalization
α, β, ϵ	Balance coefficient
$\mathcal{L}$	Objective function

The feature matrix $\mathbf{X} = \{\mathbf{x}_1; \ldots; \mathbf{x}_N\} \in \mathbb{R}^{N \times d}$ records the features of all nodes, where $\mathbf{x}_i \in \mathbb{R}^d$ is the d-dimensional feature vector associated with node v_i. The topology of the graph $\mathcal{G}$ can be represented by the adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times N}$. The notations used in this paper are summarized in Table I.

A. DMGNC

1) Masked Graph Representation Learning Module: The graph encoder and the graph decoder are denoted as $f_{\boldsymbol{\theta}_e}$ and $f_{\boldsymbol{\theta}_d}$, respectively. $\mathbf{Z} \in \mathbb{R}^{N \times d_{\mathbf{z}}}$ is the latent representation learned by the encoder. Normally, the input of general GAEs can be expressed as

$$\mathbf{Z} = f_{\boldsymbol{\theta}_e}(\mathbf{A}, \mathbf{X}). \tag{1}$$

Any type of graph neural networks can be utilized as the backbone of the encoder and decoder. For example, GCN [43] can effectively capture topological graph information, and GAT [57] can capture the importance of neighboring nodes applying an attention mechanism.

Currently, most GAE-based graph node clustering methods use a combination of reconstructed features and structures to rebuild the adjacency matrix. Inspired by the masked autoencoder, we adopt the reconstruction of the feature matrix using the adjacency matrix $\mathbf{A}$ and the partial observation node feature matrix $\tilde{\mathbf{X}}$.

Specifically, the features of the partial nodes are covered. A subset $\tilde{\mathcal{V}} \subset \mathcal{V}$ of nodes is sampled from the set of nodes, and a token [MASK] is employed to hide the features of each node in this subset to yield a learnable vector $\mathbf{x}_{[M]} \in \mathbb{R}^d$. A random sampling strategy and a large mask ratio are applied to mask the nodes, which can avoid potential bias centers and reduce redundancy in the attribute graph. After covering some of the node

features, a masked feature matrix $\tilde{\mathbf{X}} = \{\tilde{\mathbf{x}}_1; \ldots; \tilde{\mathbf{x}}_N\} \in \mathbb{R}^{N \times d}$ is obtained. The masked feature vector can be defined as

$$\tilde{\mathbf{x}}_i = \begin{cases} \mathbf{x}_{[M]}, & v_i \in \tilde{\mathcal{V}} \\ \mathbf{x}_i, & v_i \notin \tilde{\mathcal{V}}. \end{cases} \tag{2}$$

From the encoder, the latent representation $\mathbf{Z}_1$ is derived from (1).

In generative graph learning methods, either apply fewer neural networks as decoders or use a relatively simple multilayer perception. We follow [41] to utilize a single-layer graph neural network as a decoder. It enables the recovery of its input features according to the nodes themselves or even on their surrounding nodes, which is more conducive for the encoder to learn high-level latent information. In the decoder, the remask decoding technique is used to reprocess the output $\mathbf{Z}_1$ of the encoder, which means the nodes in subset $\tilde{\mathcal{V}}$ are reprocessed with another token (RMASK) with $\mathbf{z}_{[M]} \in \mathbb{R}^{d_{\mathbf{z}}}$, and the remask-processed latent representation $\mathbf{Z}_2$ is obtained. Remasked latent representation $\mathbf{Z}_2$ is specifically expressed as

$$\tilde{\mathbf{z}}_i = \begin{cases} \mathbf{z}_{[M]}, & v_i \in \tilde{\mathcal{V}} \\ \mathbf{z}_i, & v_i \notin \tilde{\mathcal{V}}. \end{cases} \tag{3}$$

To further exploit the compressed information of the data and to take advantage of the masking strategy for the clustering task, a more complete and robust final representation $\hat{\mathbf{Z}}$ is obtained by weighting the average and integrating the representation $\mathbf{Z}_1$ from the encoder and the remasked representation (RR) $\mathbf{Z}_2$

$$\hat{\mathbf{Z}} = (1 - \epsilon)\mathbf{Z}_1 + \epsilon \mathbf{Z}_2 \tag{4}$$

where the fusion coefficient ϵ is utilized to balance the weight between $\mathbf{Z}_1$ and $\mathbf{Z}_2$. In this way, the two parts of the learned latent representations are effectively integrated.

Hou et al. [41] experimentally pointed out that minimizing the mean square error loss in feature reconstruction may be difficult to optimize or may be minimized to near zero, which is not sufficient for meaningful feature reconstruction. For this reason, the scale cosine error is used to measure the reconstruction results. The reconstruction feature matrix $\tilde{\mathbf{X}}' = f_{\boldsymbol{\theta}_d}(\mathbf{A}, \mathbf{Z}_1)$ derives from the decoder, and then the feature reconstruction loss formula in the masked graph representation learning module is defined as follows:

$$\mathcal{L}_{\text{sce}} = \frac{1}{|\tilde{V}|} \sum_{v \in \tilde{V}} \left(1 - \frac{\mathbf{x}_i^{\top} \tilde{\mathbf{x}}_i}{\|\mathbf{x}_i\| \cdot \|\tilde{\mathbf{x}}_i\|}\right)^{\gamma}, \qquad \gamma \geq 1 \tag{5}$$

where γ is a hyperparameter of the scaling factor, which helps to perform better adaptation for various data.

2) Self-Optimizing Clustering Module: As graph node clustering is an unsupervised task, it cannot provide relevant feedback on whether the learned latent representations are well optimized during training due to the lack of label guidance. In this regard, to obtain more optimal clustering results and discriminative representations, we use a clustering loss (CL) $\mathcal{L}_{\text{clu}}$ to optimize the clustering task as a solution.

First, the latent representations learned by the masked graph representation learning module are input into the self-optimizing clustering module. Next, for the ith node and the jth cluster, we measure the similarity between the representation $\hat{\mathbf{z}}_i$ and the cluster centroid vector $\boldsymbol{\mu}_j$ using the t-student distribution, expressed as follows:

$$q_{ij} = \frac{(1 + \|\hat{\mathbf{z}}_i - \boldsymbol{\mu}_j\|^2 / v)^{-\frac{v+1}{2}}}{\sum_{j'=1}^{k} (1 + \|\hat{\mathbf{z}}_i - \boldsymbol{\mu}_{j'}\|^2 / v)^{-\frac{v+1}{2}}} \tag{6}$$

where v is the degree of freedom of the t-student distribution, k represents the number of cluster centers, and q_{ij} can be regarded as the soft cluster assignment of each node, i.e., the probability that node i is assigned to cluster j. We employ k-means in the initialization phase of the model to obtain the cluster and the cluster centroid vector $\boldsymbol{\mu}_j$ based on the representation learned by the autoencoder.

In addition, it is expected to optimize the data representation by learning a high-confidence distribution, which can bring the data representation closer to the cluster center and improve cohesiveness. Therefore, we derive the target distribution $\mathbf{P}$ using $\mathbf{Q}$ as follows:

$$p_{ij} = \frac{q_{ij}^2 / \sum_{i=1}^{N} q_{ij}}{\sum_{j'=1}^{k} (q_{ij'}^2 / \sum_{i=1}^{N} q_{ij'})} \tag{7}$$

where $\sum_{i=1}^{N} q_{ij}$ is the soft cluster frequency. Each assignment in $\mathbf{Q}$ is provided with a higher confidence level by normalization. The KL divergence between distributions $\mathbf{P}$ and $\mathbf{Q}$ can be minimized to facilitate the masked graph representation learning module learning a better representation of the clustering task with the following equation:

$$\mathcal{L}_{\text{clu}} = \text{KL}(\mathbf{P} \,||\, \mathbf{Q}) = \sum_{i=1}^{N} \sum_{j=1}^{k} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{8}$$

3) Joint Embedding and Clustering Optimization: The masked graph representation learning module and the self-optimizing clustering module jointly improve embedding and clustering learning and the total objective function of our framework is defined as

$$\mathcal{L} = \mathcal{L}_{\text{sce}} + \alpha \mathcal{L}_{\text{clu}} \tag{9}$$

where $\mathcal{L}_{\text{sce}}$ and $\mathcal{L}_{\text{clu}}$ are the feature reconstruction loss and the CL, respectively, and α is a hyperparameter to control the tradeoff between the two terms.

To explicitly clarify the whole process, the training procedures of DMGNC are summarized in Algorithm 1.

B. DMGNCC

Our method draws inspiration from the concept of contrastive learning proposed in [58] and [59], which learns data representations by contrasting positive and negative samples, thereby improving clustering performance. We have expanded this idea by applying contrastive learning not only at the graph node level but also by introducing an MGA and considering contrasts at the clustering level, to further enhance the accuracy and robustness of clustering. Although our work is based on the core ideas of these two preliminary studies, our research differs in objectives, methods, and application scenarios. The contrastive clustering framework we propose is an important supplement

and extension to these previous methods, aimed at solving the clustering problem of static graph data through a combination of node-level and class-level contrastive learning.

Specifically, DMGNCC consists of three jointly learned components: a PMN, a NCM, and a CCM. In PMN, we perform data augmentation on the data to construct data pairs, then apply a masked graph autocoder to extract the corresponding features, and finally perform node-level and class-level contrastive learning in NCM and CCM, respectively. These modules are described in detail as follows.

1) PMN: It can be seen from the previous research that in contrastive learning, an appropriate choice of data augmentation strategy can contribute to good performance in downstream tasks. Specifically, given a graph data $\mathcal{G}$, we perform two types of data augmentations. We randomly remove a portion of edges from the original graph at the structural level. In addition, we mask node features or add Gaussian noise at the attribute level. These operations result in the generation of two graph views, which can be expressed as $\tilde{\mathcal{G}}^a = \mathcal{T}^a(\mathcal{G})$ and $\tilde{\mathcal{G}}^b = \mathcal{T}^b(\mathcal{G})$.

Subsequently, a shared deep masked graph network $f(\cdot)$ is used to extract features from two augmented graph views via $\mathbf{H}^a = f(\tilde{\mathcal{G}}^a)$ and $\mathbf{H}^b = f(\tilde{\mathcal{G}}^b)$. This deep masked graph network is similar to the masked graph representation learning module of DMGNC. It covers the node features with a certain mask rate in the PMN with shared weights. This method can help the model learn more robust feature representations, reduce the risk of overfitting, and facilitate more accurate data clustering.

2) Node-Level Contrastive Module: The key idea of graph node contrastive learning is to exploit the differences between positive and negative data pairs to help the model learn more robust node representations, which can better handle the similarity and dissimilarity between nodes. Specifically, it encourages representations of similar pairs to be closer, while making representations of dissimilar pairs more distant. We enable the model to learn a more discriminative feature representation by maximizing the similarity between positive pairs and minimizing the similarity between negative pairs. A nonlinear multilayer perceptron $g_{S_\Theta}(\cdot)$ is utilized to map features outputs of the MGA to a subspace for node-level contrastive learning, which yields $\mathbf{Z}^a = g_{S_\Theta}(\mathbf{H}^a)$ and $\mathbf{Z}^b = g_{S_\Theta}(\mathbf{H}^b)$.

In this work, we take the features of N nodes as samples. After data augmentation, $\{\mathbf{x}_1^a, \ldots, \mathbf{x}_N^a, \mathbf{x}_1^b, \ldots, \mathbf{x}_N^b\}$ for a total of $2N$ samples can be obtained. Given a node feature $\mathbf{x}_i$, the positive pair consists of corresponding node features from two graph views $\{\mathbf{x}_i^a, \mathbf{x}_i^b\}$, naturally treating the other $2N-2$ pairs from the two views as negative pairs. We use cosine similarity to measure the pairwise similarity between samples, i.e.,

$$\cos(\mathbf{z}_i^n, \mathbf{z}_j^m) = \frac{(\mathbf{z}_i^n)^\top (\mathbf{z}_j^m)}{\|\mathbf{z}_i^n\| \|\mathbf{z}_j^m\|} \tag{10}$$

where $n, m \in \{a, b\}$, $i, j \in [1, N]$ and $\|\cdot\|$is the ℓ_2-normalization. Thereafter, for node feature $\mathbf{x}_i$, the corresponding loss can be calculated by the following equation:

$$\ell_i^a = -\log \frac{\exp(\cos(\mathbf{z}_i^a, \mathbf{z}_i^b)/\tau_1)}{\sum_{j=1}^N [\exp(\cos(\mathbf{z}_i^a, \mathbf{z}_j^a)/\tau_1) + \exp(\cos(\mathbf{z}_i^b, \mathbf{z}_j^b)/\tau_1)]} \tag{11}$$

where $\tau_1 > 0$ is a temperature parameter. Ultimately, for all nodes, the node-level contrastive loss can be calculated for each positive sample pair as follows:

$$\mathcal{L}_{\text{node}} = \frac{1}{2N} \sum_{i=1}^N (\ell_i^a + \ell_i^b). \tag{12}$$

3) Class-Level Contrastive Module: In general, the feature representation of a node is learned from its neighborhood information using unsupervised learning methods, which may not well reflect the true cluster information of the node.

In the CCM, similar to PMN, we feed the output features of the MGA into a two-layer multilayer perceptron $g_{C_\Theta}(\cdot)$. The difference, however, is that there is a softmax layer at the end, where each element of this node's feature vector can be understood as a probability of belonging to each class under this mapping via $\mathbf{Y}^a \in \mathbb{R}^{N \times C} = g_{C_\Theta}(\mathbf{H}^a)$ with C classes, and $\mathbf{Y}_{i,c}^a$ can be considered as the probability that the ith node belongs to the class c. Therefore, $\mathbf{y}_i^a$ is the ith row of $\mathbf{Y}^a$ and can be regarded as the soft label of $\mathbf{x}_i^a$ and additional feature representation, which is integrated into the node contrastive representation learning to enable the model to better distinguish different nodes and improve the accuracy of node clustering.

Likewise, we treat the $\{\mathbf{y}_i^a, \mathbf{y}_i^b\}$ of the two data-augmented graph views as a positive pair, while the remaining $2C-2$ pairs are negative. The cosine similarity is then also used to measure the similarity between these pairs

$$\cos(\mathbf{y}_i^n, \mathbf{y}_j^m) = \frac{(\mathbf{y}_i^n)^\top (\mathbf{y}_j^m)}{\|\mathbf{y}_i^n\| \|\mathbf{y}_j^m\|} \tag{13}$$

Algorithm 1 Deep Masked Graph Node Clustering

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$; number of clusters k; number of iterations T.

Output: Clustering results $\mathbf{O}$.

```
Randomly choose nodes for mask operation;
for iterator = 1 ⟶ T do
    Obtain the encoder representation Z_1 by Eq. (1);
    Generate the re-masked representation Z_2 via Eq. (3), and gain the final representation Ẑ by Eq. (4);
    Calculate the loss between the reconstructed features and the original features via Eq. (5).
    Compute the KL divergence between distribution P and Q by Eq. (8);
    Calculate the overall loss function via Eq. (9);
    Conduct the back-propagation to update parameters;
end for
return The clustering results O on distribution Ẑ.
```

Algorithm 2 Deep Masked Graph Node Contrastive Clustering

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$; number of clusters k; number of iterations T.

Output: Clustering results $\mathbf{O}$.

Generate two augmented graphs $\tilde{\mathcal{G}}^a = \mathcal{T}^a(\mathcal{G})$ and $\tilde{\mathcal{G}}^b = \mathcal{T}^b(\mathcal{G})$.
for iterator = 1 $\longrightarrow T$ **do**
  Obtain the encoder representation by $\mathbf{H}^a = f(\tilde{\mathcal{G}}^a)$ and $\mathbf{H}^b = f(\tilde{\mathcal{G}}^b)$;
  Generate the node representation via $\mathbf{Z}^a = g_{S_\Theta}(\mathbf{H}^a)$ and $\mathbf{Z}^b = g_{S_\Theta}(\mathbf{H}^b)$;
  Calculate the class representation via $\mathbf{Y}^a = g_{C_\Theta}(\mathbf{H}^a)$ and $\mathbf{Y}^b = g_{C_\Theta}(\mathbf{H}^b)$.
  Compute the node-level contrastive loss via Eq. (12);
  Calculate the class-level contrastive loss via Eq. (16);
  Compute the overall loss function via Eq. (17);
  Conduct the back-propagation to update parameters;
end for
return The clustering results $\mathbf{O}$ on distribution $\mathbf{Z}^a$ and $\mathbf{Z}^b$.

where $n, m \in \{a, b\}$, $i, j \in [1, C]$. For the node feature $\mathbf{x}_i$, the loss between classes can be calculated by the following formula:

$$\jmath_i^a = -\log \frac{\exp(\cos(\mathbf{y}_i^a, \mathbf{y}_i^b)/\tau_2)}{\sum_{j=1}^{C}[\exp(\cos\,(\mathbf{y}_i^a, \mathbf{y}_j^a)/\tau_2) + \exp(\cos(\mathbf{y}_i^b, \mathbf{y}_j^b)/\tau_2)]} \tag{14}$$

where $\tau_2 > 0$ is a temperature parameter. Furthermore, to avoid assigning a large number of nodes all to the same cluster when calculating the contrastive loss for all classes, we calculate an entropy term that spreads the predictions evenly across the clusters as follows:

$$H(\mathbf{Y}) = \sum_{i=1}^{C}[p(\mathbf{y}_i^a)\log(p(\mathbf{y}_i^a)) + p(\mathbf{y}_i^b)\log(p(\mathbf{y}_i^b))] \tag{15}$$

where $p(\mathbf{y}_i^b) = 1/N\sum_{k=1}^{N}\mathbf{Y}_{ki}^m$ is cluster assignment probability, and $k \in \{a, b\}$. Finally, we can calculate all class-level contrastive loss, i.e.,

$$\mathcal{L}_{\text{class}} = \frac{1}{2C}\sum_{i=1}^{C}(\jmath_i^a + \jmath_i^b) + \lambda H(\mathbf{Y}) \tag{16}$$

where λ is a balance coefficient.

4) Joint Contrastive Clustering Optimization: With the MGA as the core, the NCM and the CCM are jointly optimized and trained in an end-to-end model, and the overall objective function of the framework can be defined as

$$\mathcal{L} = \mathcal{L}_{\text{node}} + \beta\mathcal{L}_{\text{class}} \tag{17}$$

where β is a balance coefficient to control the tradeoff between the two terms. Eventually, nodes are mapped into a clustering space utilizing node representations learned from MGA and contrastive loss, and clustered employing a clustering layer. And the training procedures of the DMGNCC are summarized in Algorithm 2.

TABLE II
STATISTICS OF TESTED DATASETS

Dataset	# Nodes	# Features	# Edges	# Classes
ACM	3025	1870	13 128	3
Citeseer	3327	3703	4732	6
Cora	2708	1433	5429	7
Pubmed	19 717	500	44 338	3
UAI	3067	4973	28 311	19
Wiki	2405	4973	17 981	17

C. Complexity Analysis

In this work, we define the maximum dimension of the input data as d, the total number of nodes as N, the number of clusters as k, and the total number of edges as $|\mathcal{E}|$. For the DMGNC method, the computational complexity of the masking operation is $\mathcal{O}(N)$, while the complexity of the masked autoencoder module is determined to be $\mathcal{O}(Nd^2 + |\mathcal{E}|d)$. The clustering layer exhibits a time complexity of $\mathcal{O}(Nk + N\log N)$. Consequently, the aggregate computational complexity for DMGNC approximately amounts to $\mathcal{O}(Nd^2 + |\mathcal{E}|d + Nk + N\log N)$. Regarding the DMGNCC method, the complexity involved in generating augmented views is contingent on the specific operation utilized; however, for straightforward operations such as edge removal and feature masking, it typically remains at $\mathcal{O}(N)$. The computational efforts for node-level and class-level modules reach a worst-case complexity of $\mathcal{O}(N^2)$ and $\mathcal{O}(k^2)$, respectively. Since $k \ll N$, the overall computational complexity for DMGNCC is approximately $\mathcal{O}(N^2 + Nd^2 + |\mathcal{E}|d)$.

IV. EXPERIMENTS

In this section, we conduct substantial experiments to demonstrate the effectiveness of the proposed method.

A. Experimental Setup

1) Datasets: The proposed approach is evaluated on the six real-world graph datasets, as shown in Table II, and the detailed descriptions are as follows.

1) *ACM*[1] is a benchmark dataset commonly applied for evaluating the performance of graph clustering algorithms. It contains bibliographic records of articles published in the Association for Computing Machinery digital library, with each article represented as a node and the edges representing cocitation relationships between articles.
2) *Citeseer*[2] is a citation network dataset frequently used in machine learning research. It consists of scientific publications described by 3703-dimensional bag-of-words feature vectors from the field of computer science. Each article is represented by a bag-of-words feature vector, with the edges between articles representing citations.
3) *Cora*[3] is a usually utilized literature citation network dataset that consists of 2708 articles divided into seven

[1]http://dl.acm.org/
[2]http://citeseerx.ist.psu.edu/index
[3]https://relational.fit.cvut.cz/dataset/CORA

categories, each represented by a vector of 1433 lexical features. The edges of this dataset indicate citation relationships between articles, i.e., if an article cites another article, there is an edge between these two articles.

4) *Pubmed*[4] is a widely used citation relationship graph dataset, where nodes indicate scientific publications and edges indicate citation relationships. The dataset covers a subset of all research articles in the PubMed database that are biomedically relevant, including 19 717 articles, each tagged with one of three possible tags.
5) *UAI*[5] has been used in several studies as a benchmark for evaluating the performance of GCN-based methods for node classification and other tasks. Nodes representing pages are from multiple universities, with each edge representing citations.
6) *Wiki*[6] is a graph-based dataset ordinarily employed for evaluating graph embedding methods. It is derived from the English Wikipedia and consists of a directed graph with articles as nodes and hyperlinks between them as edges. The nodes are labeled with article titles, and the edges represent hyperlinks between articles.

2) Baselines: We compared our method with several state-of-the-art graph node clustering baselines to validate its effectiveness: GAE [42], MGAE [37], ARVGA [32], DAEGC [33], SDCN [45], DFCN [34], AGCN [60], EGAE [61], and DCRN [62]. The following is a brief introduction to these baselines.

1) *GAE* proposes a graph autoencoder using GCN as an encoder to learn a node embedding and uses reconstruction and regularization loss to optimize the model.
2) *MGAE* utilizes a marginalized graph autoencoder to corrupt the network node content and enable the node content to interact with the network features.
3) *ARVGA* presents an adversarial graph embedding model for graph data, and an adversarially regularized GAE to learn feature representation.
4) *DAEGC* learns a node representation by capturing the importance between neighboring nodes through an attention network in a unified framework.
5) *SDCN* introduces a structural deep clustering network that effectively combined the strengths of both autoencoder and GCN to alleviate the over-smoothing problem.
6) *DFCN* exploits a fusion module of structural and attribute information based on interdependence learning to learn a consensus node representation.
7) *AGCN* proposes an attention-driven graph clustering network considering dynamic fusion strategies and multiscale feature fusion to aggregate the multiscale features embedded.
8) *EGAE* designs a specific GAE-based graph clustering model and learns the relaxed k-means and GAE to induce the neural network to produce deep features.
9) *DCRN* utilizes a double information correlation reduction mechanism and also avoids the generation of negative samples to filter the redundant information in both views.

[4]https://pubmed.ncbi.nlm.nih.gov/download/
[5]https://github.com/zhumeiqiBUPT/AM-GCN
[6]https://github.com/thunlp/TADW

3) Parameter Setting: All parameter settings of baselines are directly utilized by the source codes provided by the authors. For our models, we use a learning rate of 0.001 with Adam optimizer and performed 300 iterations. For different datasets, we set each parameter as a uniform fixed value, i.e., $\alpha = 100$ and $\gamma = 2$ for DMGNC, node-level temperature parameters $\tau_1 = 0.5$ and $\tau_2 = 1$ for DMGNCC, respectively. In addition, parameter sensitivity analysis is conducted with respect to different values of mask ratio, fusion coefficient ϵ, scaling factor γ, and balance coefficient β for each dataset.

4) Evaluation Metric: Several evaluation metrics in the experiments to evaluate the performance of the proposed methods are introduced. We compare DMGNC and DMGNCC with several state-of-the-art graph node clustering methods. All experiments of the proposed frameworks and compared methods are run five times with means recorded as the final results with NVIDIA Tesla P100 GPU and Intel Core i5-11500 CPU.

Five well-known metrics, including accuracy (ACC), normalized mutual information (NMI), average rand index (ARI), macro F1-score (F1), and Purity, are employed to evaluate the clustering performance of all methods. Particularly, ACC is primarily employed to compare the accuracy of predicted labels against the ground truths. NMI uses mutual information to measure the similarity between two clustering results. ARI enables measuring the consistency of the clustering results with the true labels. F1 is a combination of precision rate (the ratio of the number of correct samples of clustering results to the total number of clustering results) and recall rate (the ratio between the number of correct samples of clustering results and the number of correct true samples), which measures the accuracy and confidence of the clustering. Purity is the proportion of the number of data points belonging to the most prevalent true category in each cluster to the total number of data points. For each metric, a larger value indicates a better clustering result.

B. Experimental Results

In this section, comprehensive experiments are described to evaluate different graph node clustering methods. In comparison with nine methods, the experimental results validate the superiority of the two proposed methods on six real-world graph node datasets.

1) Clustering Results: The results of node clustering are reported in Table III. From the results, we derive some beneficial observations. Compared to the baseline, our two proposed methods, DMGNC and DMGNCC, generally achieve the best performance on most of the datasets, including the smaller dataset Cora and the larger dataset Pubmed. These experimental results validate that the application of the masking strategy in graph node clustering is effective. On the one hand, DMGNC, as a generative graph node clustering method that focuses on intradata information in the graph, is advantageous to utilize RRs for clustering tasks. It is capable of providing a stronger generalization to invisible nodes. On the other hand, DMGNCC, as a contrastive graph node clustering method, can further focus on interdata information and apply contrastive ideas in MGA to pull in positive samples and push away

TABLE III
CLUSTERING PERFORMANCE (ACC %, NMI %, ARI %, F1 %, AND PURITY %) ON SIX DATASETS

Dataset/Methods		GAE	MGAE	ARVGA	DAEGC	SDCN	DFCN	AGCN	EGAE	DCRN	**DMGNC**	**DMGNCC**
ACM	ACC	84.74	87.64	78.53	88.89	89.47	90.68	90.41	88.23	90.62	**90.94**	**91.04**
	NMI	57.51	62.49	43.52	65.79	66.42	69.10	67.75	62.49	**69.34**	**69.16**	68.61
	ARI	63.95	67.14	46.90	71.55	71.61	74.55	73.74	68.37	75.02	**75.03**	**75.11**
	F1	75.03	78.17	78.44	81.10	81.17	83.26	82.70	78.94	83.11	**83.40**	**83.41**
	Purity	88.19	87.55	82.87	93.02	92.24	92.99	93.08	91.90	91.85	**93.29**	**93.35**
Citeseer	ACC	60.31	65.40	58.31	65.41	65.66	69.02	68.48	67.16	**70.65**	**71.27**	70.31
	NMI	33.90	40.50	30.83	42.33	37.91	42.13	41.14	38.71	**44.75**	**44.40**	43.90
	ARI	33.24	40.84	30.08	43.17	37.93	44.00	43.46	40.56	46.22	**46.55**	**46.36**
	F1	44.67	51.24	55.24	55.33	49.34	52.63	52.67	51.05	55.54	**56.70**	**56.36**
	Purity	66.62	67.15	64.41	73.08	67.65	71.39	70.61	73.43	72.08	**73.71**	**73.10**
Cora	ACC	57.42	63.54	64.54	71.73	53.91	67.36	64.36	72.45	68.13	**73.12**	**72.76**
	NMI	42.06	45.63	46.51	54.84	35.23	52.60	46.07	52.88	53.97	**54.80**	**54.73**
	ARI	34.29	38.12	39.15	49.16	27.58	44.78	38.53	50.33	46.15	**51.29**	**50.97**
	F1	45.13	48.08	53.09	57.57	42.23	57.69	53.81	58.14	58.01	**58.45**	**59.30**
	Purity	66.39	68.85	68.50	**76.12**	56.45	71.66	66.77	76.11	74.85	75.22	**76.87**
Pubmed	ACC	60.53	42.75	58.91	66.25	63.08	68.33	64.71	69.41	69.78	**70.46**	**70.62**
	NMI	23.29	7.84	20.33	26.33	22.27	28.89	26.21	30.04	32.02	**34.21**	**34.65**
	ARI	22.15	3.32	20.52	26.48	21.44	29.27	24.16	31.25	31.35	**33.57**	**33.72**
	F1	47.90	42.08	51.20	53.57	49.11	56.92	52.78	55.73	57.45	**58.44**	**57.82**
	Purity	80.12	85.48	79.19	88.53	83.93	88.10	84.10	82.64	87.15	**89.11**	**88.94**
UAI	ACC	30.10	41.47	20.72	38.15	26.07	37.89	40.81	42.04	39.65	**43.35**	**44.28**
	NMI	25.86	40.23	18.38	38.62	20.83	37.52	35.91	**41.43**	40.84	40.90	**41.98**
	ARI	8.51	22.97	5.41	22.74	7.64	17.23	18.80	23.43	20.97	**23.51**	**23.98**
	F1	17.02	29.31	16.56	29.12	19.01	29.21	32.79	28.68	**34.57**	30.27	**32.32**
	Purity	35.02	38.11	27.46	47.95	28.13	48.80	43.01	**52.60**	47.21	**49.12**	47.47
Wiki	ACC	31.10	50.28	27.44	38.14	41.63	47.98	39.22	50.35	48.42	**51.43**	**52.72**
	NMI	26.22	**48.32**	20.31	31.41	41.53	45.69	39.12	46.13	45.88	**48.76**	47.11
	ARI	15.54	**34.72**	7.85	17.91	16.48	16.15	15.15	32.05	27.29	33.34	**33.62**
	F1	25.66	**40.15**	20.23	24.54	26.87	27.41	36.83	37.24	**37.40**	35.21	35.52
	Purity	36.38	56.15	34.69	46.91	50.36	56.92	48.02	**60.96**	57.89	**58.25**	58.01

Note: Best and runner-up results are highlighted in bold **red** and **blue**, respectively.

negative ones to improve feature discrimination. DMGNC and DMGNCC have their advantages in performance on different datasets and achieve better performance than other state-of-the-art graph node clustering methods on most of the datasets. It is worth mentioning that some of the metrics of the two proposed methods on multiclass datasets (Wiki and UAI) may lag slightly behind the other methods. We consider that it may be that the data are possibly harder to discriminate in data with multiple classes after the masked operation performed by the MGA. This part will be further investigated in our subsequent work. It can also be seen that the performance of DAEGC and AGCN based on the attention mechanism performs well on most of the datasets, which shows that the graph attention network may be more expressive in node clustering Therefore, GAT is employed as the backbone network for both the MGA and the decoder in our models.

It is worth noting that the differing performances can largely be attributed to the inherent characteristics of each dataset, such as size, complexity, feature space, and noise levels, which may inherently favor either a generative method such as DMGNC or a contrastive version such as DMGNCC. DMGNCC employs contrastive learning to enhance feature representations by distinguishing between positive and negative node pairs, a strategy whose effectiveness is contingent on the dataset's quality and distribution. While contrastive learning may significantly improve clustering by capturing nuanced differences between nodes in some datasets, its advantages might not be as pronounced in others, particularly where the graph structure is less informative or more noise pervasive. These methodological differences and design choices between DMGNC and DMGNCC, including DMGNC's focus on feature reconstruction and DMGNCC's emphasis on node relationship contrast, are pivotal in understanding their dataset-dependent performance. Moving forward, we suggest exploring hybrid models that amalgamate the strengths of both methods and adjusting the DMGNCC framework to better accommodate less conducive datasets.

2) Stability Experiment: Fig. 4 records the loss values of DMGNC and DMGNCC for different iteration epochs on different datasets. It can be observed that as the number of iterations increases, the loss values for the six datasets gradually decrease and gradually converge. Eventually, it converges to a stable value with a slight fluctuation when the number of epochs is large enough, which indicates convergence. It can be

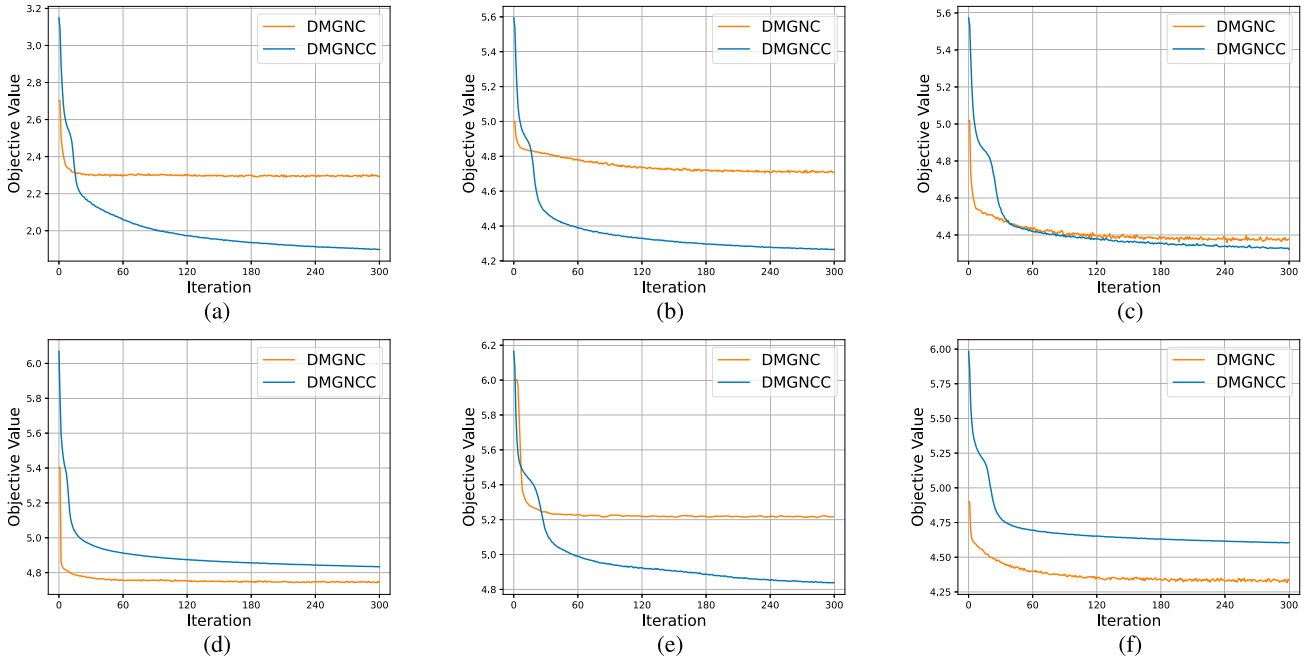

Fig. 4. Convergent curves of the proposed frameworks DMGNC and DMGNCC on the six datasets. (a) ACM. (b) Citeseer. (c) Cora. (d) Pubmed. (e) UAI. (f) Wiki.

discerned that both of our proposed methods are capable of stable training at a certain number of iterations and that performing masked feature reconstruction facilitates robust training of the node representation.

To demonstrate that our method can effectively integrate information under the prevalent noise in the real-world scenario and the performance of our method in the presence of contaminated node features, we sampled random noise from a Gaussian distribution $\mathcal{N}(1, \sigma^2)$ and added it to the input data, where σ^2 varied between 0.1 and 0.9, and evaluating the performance of various advanced deep graph node clustering methods. The experimental results are shown in Fig. 6. By adding noise at different degrees, compared to other advanced deep graph node clustering methods, our method has better noise resistance and is more robust.

3) Visualization Results: To intuitively show the clustering performance, Fig. 5 employs the t-SNE to present scatter diagrams of different methods on Cora. It can be observed from the figure that both of our proposed methods, DMGNC and DMGNCC, are able to better cluster the seven classes in the Cora data together, visually demonstrating that having better clustering.

C. Parameter Study

In this section, to investigate the performance variation and the effectiveness of the parameter settings for the two proposed methods, DMGNC and DMGNCC, under different settings, the following parameter sensitivity analysis is performed.

The parameter sensitivity analysis for DMGNC is shown in Fig. 7 and for DMGNCC in Fig. 8. Both Figs. 7(a) and 8(a) clearly show the variation curves of the performance of the six datasets with different mask ratios. It can be seen from the figure that when the mask ratio is low, the node feature reconstruction does not provide more assistance for learning effective embedding features, and the accuracy does not meet expectations. When the mask ratio is large, it is difficult to provide more effective information for subsequent tasks due to the loss of too much node information. Therefore, we set the mask ratio between 0.4 and 0.6 for different datasets in our experiments.

Fig. 7(b) shows the performance variation curves of the six datasets with different fusion coefficients ϵ. It can be shown that the RRs can provide varying degrees of effectiveness for clustering tasks. But the closer the fusion coefficient ϵ is to 1, the larger the weight of $\mathbf{Z}_2$ in the final representation used for clustering, resulting in excessive information loss and degraded clustering performance. Therefore, their weights should be controlled to a reasonable degree.

Fig. 7(c) records the performance variation curves for different scaling factors γ for the six datasets. We observe that the accuracy of the clustering can be improved to some extent when γ is taken as either 2 or 3, providing further demonstration that the usage of scaled cosine error can further improve performance.

Fig. 8(b) illustrates the variation curves of performance for different balance coefficients β on the six datasets. As can be viewed from the figure, the loss of both contrastive learning modules provides varying degrees of contribution to the clustering task. However, the performance degrades when each component is over-represented, indicating that it is difficult to take better advantage of the MGA with node-level loss or class-level loss alone. Therefore, their weights should also be kept within a sensible degree.

Fig. 8(c) shows the cross-entropy balance coefficient in the class-level loss. As seen from the figure, the inclusion of cross

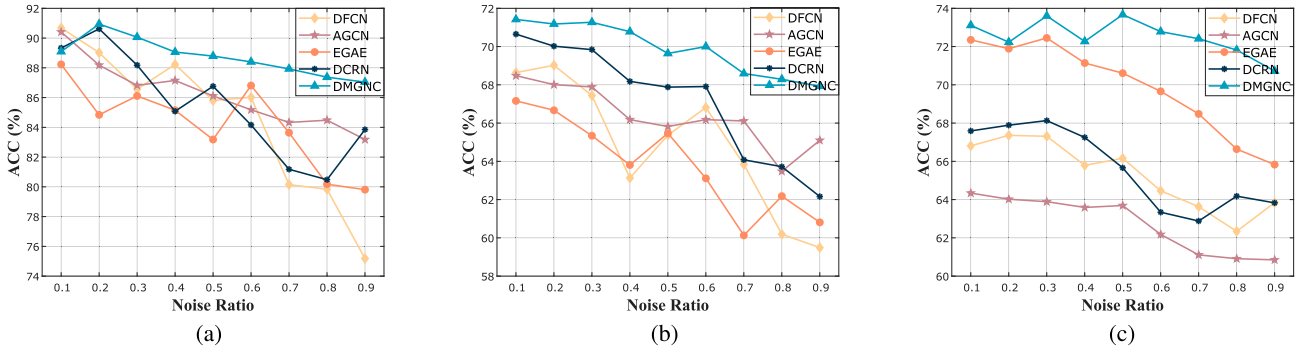

Fig. 5. Clustering accuracy for (a) ACM; (b) Citeseer; and (c) Cora datasets with different noise ratios.

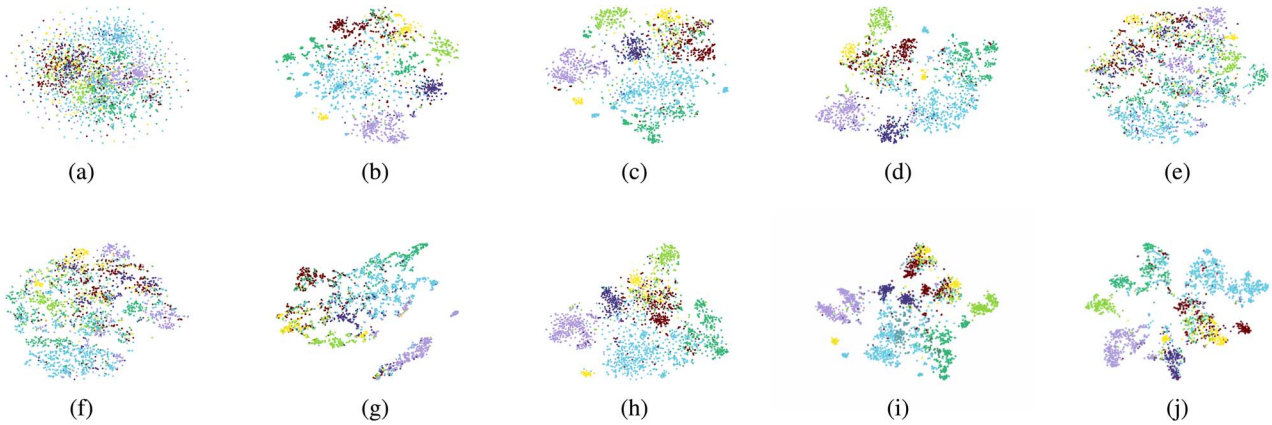

Fig. 6. t-SNE visualizations of different methods on Cora, where each color corresponds to one class. (a) Raw Data. (b) GAE. (c) MGAE. (d) ARVGA. (e) SDCN. (f) AGCN. (g) DFCN. (h) DCRN. (i) DMGNC. (j) DMGNCC.

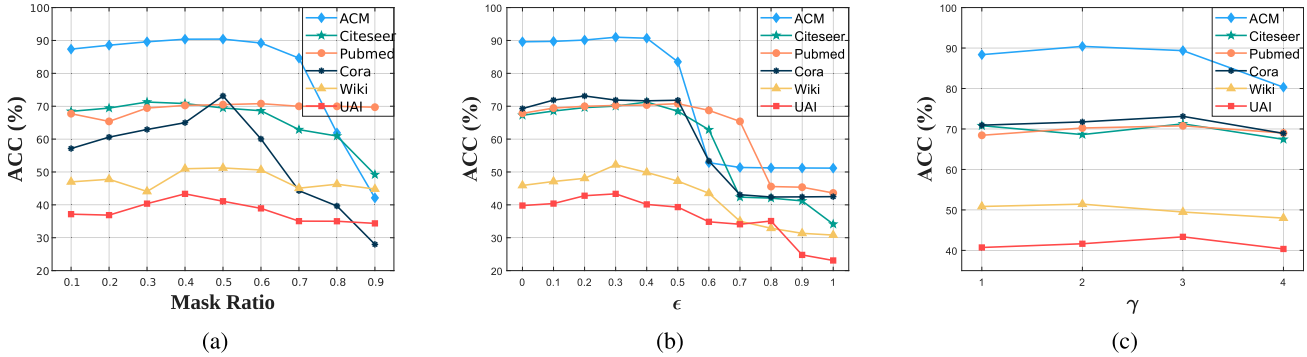

Fig. 7. Clustering accuracy of DMGNC with different (a) mask ratios; (b) fusion coefficient ϵ; and (c) scaling factor γ.

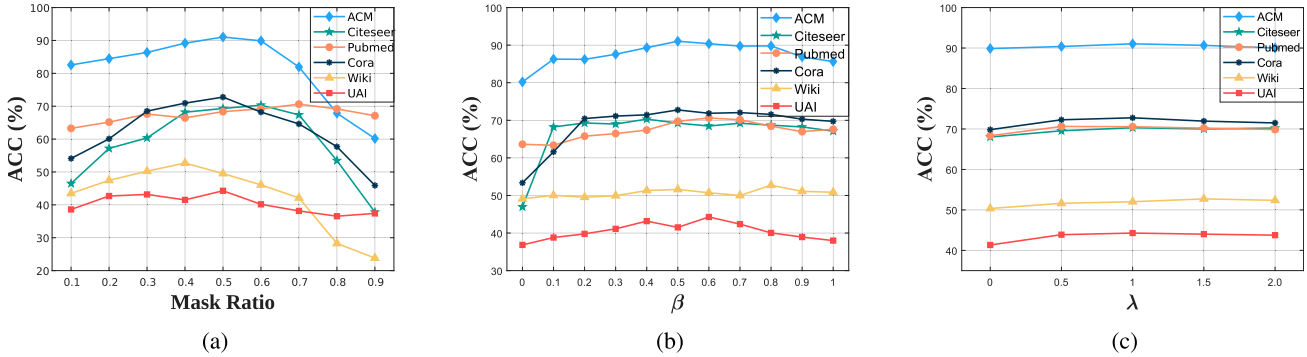

Fig. 8. Clustering accuracy of DMGNCC with different (a) mask ratios; (b) fusion coefficient β; and (c) balance coefficient λ.

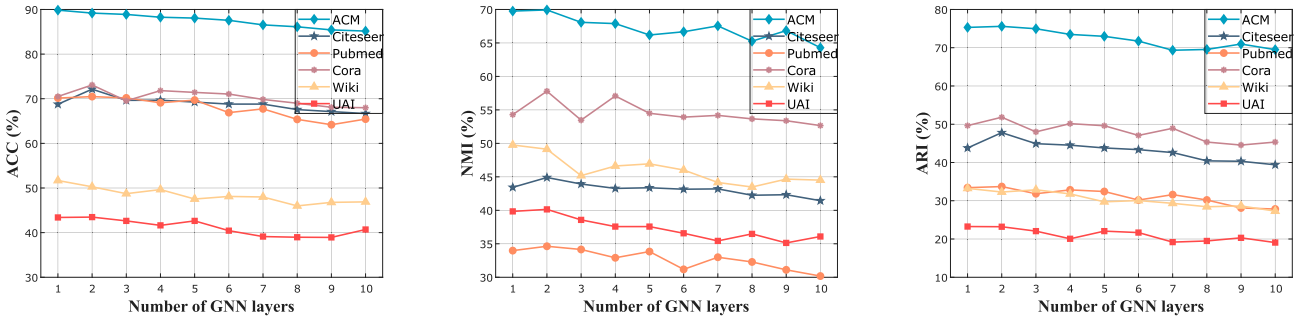

Fig. 9. Clustering performance (a) ACC. (b) NMI. (c) ARI of six datasets with different number of GNN hidden layers.

TABLE IV
ABLATION STUDY OF OUR PROPOSED MODULES DMGNC AND DMGNCC

	DMGNC					DMGNCC			
Dataset/Module	MGA	✓	✓	✓	✓	PMN	✓	✓	✓
	CL	×	✓	×	✓	NCM	✓	×	✓
	RR	×	×	✓	✓	CCM	×	✓	✓
ACM	ACC	86.50	89.59	89.88	**90.94**	ACC	85.62	80.21	**91.04**
	NMI	63.48	65.51	66.83	**69.16**	NMI	57.52	48.24	**68.61**
	ARI	70.21	71.62	72.35	**75.03**	ARI	62.71	51.24	**75.11**
	F1	79.21	81.13	81.62	**83.40**	F1	75.20	67.51	**83.41**
Citeseer	ACC	68.65	68.94	70.78	**71.27**	ACC	67.12	46.98	**70.31**
	NMI	43.28	43.92	44.14	**44.40**	NMI	40.57	23.44	**43.90**
	ARI	43.54	44.01	45.59	**46.55**	ARI	41.74	19.06	**46.36**
	F1	53.80	54.31	55.93	**56.70**	F1	41.70	33.09	**56.36**
Cora	ACC	67.49	69.28	70.02	**73.12**	ACC	69.72	53.36	**72.76**
	NMI	53.92	54.21	54.41	**54.80**	NMI	51.21	35.97	**54.73**
	ARI	44.66	44.78	45.06	**51.29**	ARI	46.60	25.37	**50.97**
	F1	52.49	54.70	54.92	**58.45**	F1	55.34	39.20	**59.30**

Note: Bold indicates best results.

entropy in each dataset is capable of improving the clustering performance to some extent, indicating that this term can effectively prevent the network from assigning all results to one class. In our experiments, we uniformly set λ to 1 for each dataset.

Furthermore, to verify that our method can effectively avoid the over-smoothing phenomenon, we performed a sensitivity analysis of the GNN hidden layers for all datasets and explored the clustering performance with the number of GNN layers varying from 1 to 10. As can be seen in Fig. 9, our method is able to show stable performance at different numbers of GNN layers, effectively avoiding the phenomenon of over-smoothing.

D. Ablation Study

In this section, we describe ablation experiments on the proposed methods, DMGNC and DMGNCC, to verify their efficiency and effectiveness.

For DMGNC, we stack the MGA, CL, and RR to conduct ablation experiments on ACM, Citeseer, and Cora gradually. In detail, we utilize the mask graph autoencoder to verify the validity. Then, the CL and RR are added to the model separately. Finally, the three components are combined into the whole framework, which is our overall complete framework DMGNC to perform the clustering task. It enables a clear measurement of the impact of each component. From Table IV, we observe that when stacking the modules one by one, every metric employed to evaluate the clustering performance increases, indicating that each module of DMGNC contributes to the performance improvement of the clustering task.

For DMGNCC, we applied the MGA to build the PMN and then stacked NCM and CCM, respectively, for progressive ablation experiments on ACM, Citeseer, and Cora. Specifically, the node-level contrast and the class-level contrast were performed in PMN, respectively, as well as the full network framework for the final execution. From Table IV, we similarly observe that each metric used to assess clustering performance increases when stacking modules one by one. This demonstrates that the two contrastive modules of DMGNCC can gather more information from within nodes and classes, which validates that each module of DMGNCC is able to improve the performance of the clustering task.

Therefore, this ablation study validates the effectiveness of each module in DMGNC and DMGNCC.

V. Conclusion

In this article, we proposed a clustering method called DMGNC and an extended version named DMGNCC for deep graph node clustering using an MGA. DMGNC jointly performed feature reconstruction learning graph embedding and graph clustering in a unified framework and used the masked latent representation for the clustering task to further exploit the valid information of the masked feature reconstruction through the masking strategy. DMGNCC utilized positive and negative node pairs in a network of MGA for node-level and class-level contrastive learning. Both methods of masked feature reconstruction are capable of robust training that favors node representation. Extensive experiments on deep graph node clustering baselines demonstrate the effectiveness of our proposed methods. This article focuses on methodological aspects and the general applicability to benchmark datasets. In the future, we will dedicate ourselves to further exploring more effective and efficient methods, and to investigating specific domain applications such as bioinformatics, chemical molecules, or recommendation systems.

References

[1] W. Hu et al., "Open graph benchmark: Datasets for machine learning on graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1–16.

[2] F. Nie, L. Tian, and X. Li, "Multiview clustering via adaptively weighted procrustes," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 2022–2030.

[3] K. Liu, F. Xue, X. He, D. Guo, and R. Hong, "Joint multi-grained popularity-aware graph convolution collaborative filtering for recommendation," *IEEE Trans. Comput. Social Syst.*, vol. 10, no. 1, pp. 72–83, Feb. 2023.

[4] Y. Xie, S. Tong, P. Zhou, Y. Li, and D. Feng, "Efficient storage management for social network events based on clustering and hot/cold data classification," *IEEE Trans. Comput. Social Syst.*, vol. 10, no. 1, pp. 120–130, Feb. 2023.

[5] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *J. Roy. Statist. Soc. C*, vol. 28, no. 1, pp. 100–108, 1979.

[6] L. Zhang, L. Fu, T. Wang, C. Chen, and C. Zhang, "Mutual information-driven multi-view clustering," in *Proc. 32nd ACM Int. Conf. Inf. Knowl. Manage.*, 2023, pp. 3268–3277.

[7] J. Cai, S. Wang, C. Xu, and W. Guo, "Unsupervised deep clustering via contractive feature representation and focal loss," *Pattern Recognit.*, vol. 123, 2022, Art. no. 108386.

[8] J. Cai, S. Wang, and W. Guo, "Unsupervised embedded feature learning for deep clustering with stacked sparse auto-encoder," *Expert Syst. Appl.*, vol. 186, 2021, Art. no. 115729.

[9] J. Cai, Y. Zhang, S. Wang, J. Fan, and W. Guo, "Wasserstein embedding learning for deep clustering: A generative approach," *IEEE Trans. Multimedia*, 2024.

[10] M. Ester et al., "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 226–231.

[11] L. Fu, Z. Chen, Y. Chen, and S. Wang, "Unified low-rank tensor learning and spectral embedding for multi-view subspace clustering," *IEEE Trans. Multimedia*, vol. 26, pp. 7567–7580, 2024.

[12] J. Cai, J. Fan, W. Guo, S. Wang, Y. Zhang, and Z. Zhang, "Efficient deep embedded subspace clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 21–30.

[13] H.-J. Li, Y. Feng, C. Xia, and J. Cao, "Overlapping graph clustering in attributed networks via generalized cluster potential game," *ACM Trans. Knowl. Discovery Data*, vol. 18, no. 1, pp. 1–26, 2023.

[14] H. Li, W. Xu, C. Qiu, and J. Pei, "Fast Markov clustering algorithm based on belief dynamics," *IEEE Trans. Cybern.*, vol. 53, no. 6, pp. 3716–3725, Jun. 2023.

[15] H. Li, H. Cao, Y. Feng, X. Li, and J. Pei, "Optimization of graph clustering inspired by dynamic belief systems," *IEEE Trans. Knowl. Data Eng.*, early access, Nov. 29, 2023.

[16] S. Wang, L. Fu, Z. Wang, H. Xu, and W. Zhu, "Multigraph random walk for joint learning of multiview clustering and semisupervised classification," *IEEE Trans. Comput. Social Syst.*, vol. 9, no. 3, pp. 926–939, Jun. 2022.

[17] S. Huang, Y. Zhang, L. Fu, and S. Wang, "Learnable multi-view matrix factorization with graph embedding and flexible loss," *IEEE Trans. Multimedia*, vol. 25, pp. 3259–3272, 2023.

[18] Y. Liu et al., "Simple contrastive graph clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 27, 2023.

[19] Y. Liu et al., "Dink-Net: Neural clustering on large graphs," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2023, pp. 21794–21812.

[20] Y. Liu et al., "Reinforcement graph clustering with unknown cluster number," in *Proc. 31st ACM Int. Conf. Multimedia*, 2023, pp. 3528–3537.

[21] J. Cai, Y. Han, W. Guo, and J. Fan, "Deep graph-level clustering using pseudo-label-guided mutual information maximization network," *Neural Comput. Appl.*, vol. 36, no. 16, pp. 9551–9566, 2024.

[22] J. Cai, W. Guo, and J. Fan, "Unsupervised deep discriminant analysis based clustering," 2022, *arXiv:2206.04686*.

[23] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 793–803.

[24] Q. Zhu, C. Yang, Y. Xu, H. Wang, C. Zhang, and J. Han, "Transfer learning of graph neural networks with ego-graph information maximization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 1766–1779.

[25] Z. Wu, Z. Zhang, and J. Fan, "Graph convolutional kernel machine versus graph convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, pp. 1–23, 2024.

[26] Z. Wu, Z. Chen, S. Du, S. Huang, and S. Wang, "Graph convolutional network with elastic topology," *Pattern Recognit.*, vol. 151, 2024, Art. no. 110364.

[27] Z. Wu, X. Lin, Z. Lin, Z. Chen, Y. Bai, and S. Wang, "Interpretable graph convolutional network for multi-view semi-supervised learning," *IEEE Trans. Multimedia*, vol. 25, pp. 8593–8606, 2023.

[28] F. Tian, B. Gao, Q. Cui, E. Chen, and T. Liu, "Learning deep representations for graph clustering," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1293–1299.

[29] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 478–487.

[30] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 3438–3445.

[31] S. Wang, J. Yang, J. Yao, Y. Bai, and W. Zhu, "An overview of advanced deep graph node clustering," *IEEE Trans. Comput. Social Syst.*, vol. 11, no. 1, pp. 1302–1314, Feb. 2024.

[32] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 2609–2615.

[33] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, "Attributed graph clustering: A deep attentional embedding approach," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 3670–3676.

[34] W. Tu et al., "Deep fusion clustering network," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 9978–9987.

[35] 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.

[36] H. Zhao, X. Yang, Z. Wang, E. Yang, and C. Deng, "Graph debiased contrastive learning with joint representation clustering," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, 2021, pp. 3434–3440.

[37] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "MGAE: Marginalized graph autoencoder for graph clustering," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 889–898.

[38] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. B. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 15979–15988.

[39] H. Bao, L. Dong, S. Piao, and F. Wei, "BEiT: BERT pre-training of image transformers," in *Proc. 10th Int. Conf. Learn. Representations*, 2022, pp. 1–18.

[40] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.*, 2019, pp. 4171–4186.
[41] Z. Hou et al., "GraphMAE: Self-supervised masked graph autoencoders," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, 2022, pp. 594–604.
[42] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*.
[43] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Representations*, 2017, pp. 1–14.
[44] J. Park, M. Lee, H. J. Chang, K. Lee, and J. Y. Choi, "Symmetric graph convolutional autoencoder for unsupervised graph representation learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6519–6528.
[45] 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.
[46] X. He, B. Wang, Y. Hu, J. Gao, Y. Sun, and B. Yin, "Parallelly adaptive graph convolutional clustering model," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 4451–4464, Apr. 2024.
[47] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607.
[48] X. Wang, D. Zhang, H. Tan, and D. Lee, "A self-fusion network based on contrastive learning for group emotion recognition," *IEEE Trans. Comput. Social Syst.*, vol. 10, no. 2, pp. 458–469, Apr. 2023.
[49] 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. 1–12.
[50] P. Velicković, W. Fedus, W. L. Hamilton, P. Lio, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *Proc. 7th Int. Conf. Learn. Representations*, 2019, pp. 1–17.
[51] R. D. Hjelm et al., "Learning deep representations by mutual information estimation and maximization," in *Proc. 7th Int. Conf. Learn. Representations*, 2019, pp. 1–24.
[52] K. Hassani and A. H. K. Ahmadi, "Contrastive multi-view representation learning on graphs," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 4116–4126.
[53] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep graph contrastive representation learning," 2020, *arXiv:2006.04131*.
[54] W. Xia, Q. Gao, M. Yang, and X. Gao, "Self-supervised contrastive attributed graph clustering," 2021, *arXiv:2110.08264*.
[55] Y. Liu, Y. Zheng, D. Zhang, H. Chen, H. Peng, and S. Pan, "Towards unsupervised deep graph structure learning," in *Proc. ACM Web Conf.*, 2022, pp. 1392–1403.
[56] F. Devvrit, A. Sinha, I. Dhillon, and P. Jain, "S3GC: Scalable self-supervised graph clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 3248–3261.
[57] P. Velicković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Representations*, 2017, pp. 1–12.
[58] Y. Li, P. Hu, Z. Liu, D. Peng, J. T. Zhou, and X. Peng, "Contrastive clustering," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 8547–8555.
[59] Y. Li, M. Yang, D. Peng, T. Li, J. Huang, and X. Peng, "Twin contrastive learning for online clustering," *Int. J. Comput. Vis.*, vol. 130, no. 9, pp. 2205–2221, 2022.
[60] 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.
[61] 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.
[62] Y. Liu et al., "Deep graph clustering via dual correlation reduction," in *Proc. 36th AAAI Conf. Artif. Intell.*, 2022, pp. 7603–7611.

Jinbin Yang received the B.S. degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China, in 2021, where he is currently working toward the M.S. degree with the College of Computer and Data Science, Fuzhou University, Fuzhou, China.

His research interests include deep clustering, deep learning, and graph neural networks.

Jinyu Cai received the Ph.D. degree in computer science and technology from the College of Computer and Data Science, Fuzhou University, Fuzhou, China, in 2023.

He is currently a Postdoc Research Fellow with the Institute of Data Science, National University of Singapore, Singapore. His research interests include machine learning, computer vision, and pattern recognition.

Luying Zhong received the B.S. degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China, in 2021, where she is currently working toward the Ph.D. degree with the College of Computer and Data Science, Fuzhou University.

Her research interests include edge computing, federated learning, graph learning, and graph neural networks.

Yueyang Pi received the B.S. degree from the College of Computer Science, Hunan University of Technology, in 2021. He is currently working toward the Ph.D. degree with the College of Computer and Data Science, Fuzhou University, Fuzhou, China.

His research interests include machine learning, active learning, and optimization.

Shiping Wang (Senior Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2014.

He is currently a Professor with the College of Computer and Data Science, Fuzhou University, Fuzhou, China, and the Director of the Key Laboratory of Intelligent Metro, Fujian Provincial University, Fuzhou, China. He was a Visiting Scholar with the University of Alberta, Edmonton, AB, Canada, from 2013 to 2014. He worked as a Research Assistant with the National University of Singapore, Singapore, from 2014 to 2014, and a Research Fellow with Nanyang Technological University, Singapore, from 2015 to 2016. He was also a Visiting Researcher with Peking University, Beijing, China, from 2019 to 2020. His research interests include machine learning, deep learning, feature representation, and multimodal fusion.