Knowledge-Based Systems 279 (2023) 110961

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

DCOM-GNN: A Deep Clustering Optimization Method for Graph Neural Networks

Haoran Yang [*], Junli Wang, Rui Duan, Chungang Yan

Key Laboratory of Embedded System and Service Computing, Ministry of Education, Shanghai 201804, China
National (Province-Ministry Joint) Collaborative Innovation Center for Financial Network Security, Tongji University, Shanghai 201804, China

ARTICLE INFO

Article history:
Received 8 June 2023
Received in revised form 30 August 2023
Accepted 31 August 2023
Available online 9 September 2023

Keywords:
Deep clustering
Graph neural networks
Inter-cluster distance
Intra-cluster distance
Optimization method

ABSTRACT

Deep clustering plays an important role in data analysis, and with the prevalence of graph data nowadays, various deep clustering models on graph are constantly proposed. However, due to the lack of more adequate clustering guidance, the discriminability of feature representation learned from these models for the clustering task is limited. Therefore, for the purpose of enabling the output of these models to be more cluster-oriented, we propose a Deep Clustering Optimization Method for Graph Neural Networks (DCOM-GNN), which can be attached to the original model architecture conveniently. For DCOM-GNN, it contains two components, one is the inter-cluster distance optimization module, whose role is to further adjust the distance between clusters of the original model output rationally. Another one is the intra-cluster distance optimization module, which aims to improve the cohesiveness of the original model output. Comprehensive experiments show that the performance of various deep clustering models on graph can be significantly improved after adding DCOM-GNN.

1. Introduction

As an important data analysis method, the purpose of clustering is to reveal the intrinsic nature of the samples and the rule of interconnection. With the development of deep learning, deep clustering [1–3] that can be considered as the combination of representation learning and clustering objective has emerged. Compared with traditional clustering algorithms (e.g., K-means [4] and spectral clustering [5]), deep clustering is significantly superior in handling large-scale, high-dimensional datasets. Initially, the deep clustering models represented by autoencoder-based only take into account the data itself [6–8].

In recent years, massive amounts of data formed by the graph have been generated in various real-world fields, such as data in social network [9,10], citation network [11,12], protein network [13,14], financial network [15]. For these graph data, there is rich structural information in addition to the data itself, which obviously is also quite important should not be neglected. Thus, for the purpose of mining more valuable information from graph data, deep clustering models on graph [16–18] are gradually attracting the attention of researchers. These models take full advantage of graph neural networks (GNNs) (e.g., Graph Convolutional Networks [19], Graph Attention Networks [20], Graph Contrastive Learning [21]), and transform graph data into the low-dimensional feature representation, while attempting to retain the properties of these data in vector space as much as possible. Subsequently, in cooperation with the respective clustering objective function, the finally clustering-oriented feature representation is obtained. For instance, a GCN-based clustering method is proposed in [22], which contains a stochastic multi-clustering framework that can effectively improve the computational efficiency. Peng et al. propose an attention-based deep clustering model on graph [23], which consists of two modules to implement feature fusion from layer-wise as well as scale-wise respectively, considering both local and global information in the process of clustering. In order to capture higher-order neighbor information to further improve the clustering performance, an adaptive graph convolution model (GCA) is proposed in [24], which is trained with spectral clustering based on the acquisition of the smooth feature representation.

However, even though some improvements have been achieved for deep clustering models on graph, the discrimination of the feature representation learned from these models for the clustering task remains unsatisfactory, i.e., the cluster guidance received by the model is still inadequate, which result in limited clustering performance of the model. Considering that the clustering task pursues "internal denseness and external sparseness", i.e., the distances between samples in the same cluster (the intra-cluster distance) should be as close as possible, and the distances between samples in different clusters (the inter-cluster distance) should be as far away as possible within a

[*] Corresponding author at: National (Province-Ministry Joint) Collaborative Innovation Center for Financial Network Security, Tongji University, Shanghai 201804, China.
E-mail address: 2010498@tongji.edu.cn (H. Yang).

https://doi.org/10.1016/j.knosys.2023.110961
0950-7051/© 2023 Elsevier B.V. All rights reserved.

reasonable range according to the actual situation. In other words, regardless of the viewpoints from which deep clustering models implement their guidance for the clustering task, there is one situation in which they are all consistent. Thus, if we enable further rationalization for the inter-cluster distance and reduction for the intra-cluster distance of the feature representation learned by original deep clustering models on graph, i.e., enhance the strength of these models' clustering-oriented guidance based on their existing, then the goal of improving their clustering performance is potentially achievable.

Therefore, in this paper, we propose a Deep Clustering Optimization Method for Graph Neural Networks (DCOM-GNN), which can be conveniently attached to the architecture of existing deep clustering models on graph and enable these models to receive more adequate clustering-oriented guidance. For DCOM-GNN, it consists of an inter-cluster distance optimization module and an intra-cluster distance optimization module. The purpose of the first module is to adjust the inter-cluster distance of the original model output more rationally, which assigns corresponding weight coefficients to clusters according to the similarities between them. In this process, we consider the attribute and the structural similarity to distinguish the degree of influence between different clusters. Next, with the aim of improving the cohesiveness of the original model output, we design the second module, which imposes corresponding "gravitation" on each node in the cluster based on the distance between nodes from the centroid in the same cluster, so as to make the intra-cluster distance more compact. Finally, benefiting from the DCOM-GNN integration into the original model and playing a corrective role in every epoch iteration during the training process, the model enables learning more discriminative feature representation for the clustering task compared with the previous one, while in order to avoid the case that the clustering performance of the model decreases instead of increases due to over-correction caused by DCOM-GNN, we also include the moderation of the original model output in the final output. Extensive experiments demonstrate that adding DCOM-GNN can improve the performance of various deep clustering models on graph effectively.

Overall, our major contributions can be summarized as below:

(1) Considering that most existing deep clustering models on graph with regard to clustering-oriented guidance are still inadequate, for the sake of enabling the feature representation learned from the original model more discriminative towards clustering task, we propose a Deep Clustering Optimization Method for Graph Neural Networks (DCOM-GNN).

(2) For DCOM-GNN, we first design an inter-cluster distance optimization module, which assigns corresponding weighting coefficients to clusters based on similarity at both the attribute and structure between them. In addition, we design another intra-cluster distance optimization module, so that the distance of nodes within the same cluster becomes more compact.

(3) We add DCOM-GNN to some representative deep clustering models on graph proposed in recent years, and the experimental results show that DCOM-GNN improves the performance of these models significantly, which demonstrates its effectiveness.

The rest of this paper is organized as follows: In Section 2, we introduce the work related to DCOM-GNN. Followed by Section 3, we describe the two modules that constitute the DCOM-GNN in detail. Then in Section 4, we show the performance after adding DCOM-GNN to some representative deep clustering models on graph and conduct relevant analysis. Finally, we conclude this work in Section 5.

2. Related work

2.1. Traditional deep clustering models

The initial deep clustering models ignore the rich structural information between the data, and focus only on extracting the features of the data itself. For instance, Deep Embedded Clustering (DEC) as a classical deep clustering model is proposed in [25], which determines sample belonging clusters by the Student's t-distribution [26] and an auxiliary distribution. Guo et al. argue that DEC ignores the preservation of data attributes, which may lead to the corruption of the vector space. Thus, they propose a method called deep convolutional embedded clustering in [27]. Similarly, considering that DEC also cannot ensure the preservation of the local structure, a method to improve local structure preservation is proposed in [28], which achieves the preservation of data structure features by fusing the clustering loss and loss in the autoencoder. In addition, following the inspiration of DEC, there are some other methods [29–31] that have been proposed.

2.2. Deep clustering models on graph

In recent years, motivated by the impressive achievements of many GNNs-based methods in encoding the graph structure, some deep clustering models on graph have been proposed. Compared with the previous models, they can integrate structural information into the process of deep clustering.

Most of such models are dedicated to learning more valuable feature representation, and consider clustering as one of the downstream tasks. The initial work of deep clustering on graphs and their models are relatively simple, but they do not take into account the multiple views and different degrees of influence among node neighbors, so the various information implied in the feature representation they learn is relatively insufficient. For instance, Variational Graph Auto-Encoders (VGAE) [32] migrates Variational Auto-Encoders to the graph, and utilizes GCN as an encoder to integrate graph structure information into the feature representation learning process. Hu et al. [33] propose a deep graph clustering method in social networks, which discovers clusters by considering node correlations and integrating the content interaction of nodes into the graph learning process. A more powerful encoding mechanism is able to improve the graph representation learning ability of the model, thus compared with the model in [32], the method in [34] can learn better feature representation due to the presence of adversarial processes. Similarly, the model in EGAE [35] utilizes a multilayer GCN to learn the feature representation of nodes, which is achieved on the basis of obtaining the similarity between the nodes by computing their inner product. Furthermore, some researchers attempt to mine more potential structural information that makes the learned feature representation more enrichment, and some models are proposed based on this. As a representative of them, MAGCN [36] leverages GCN to reconstruct feature representation and common relationship graphs. Meanwhile, a multi-view attribute graph attention mechanism is designed in MAGCN to reduce noise, thus more informative feature representation can be learned from MAGCN. A GAT-based method for hierarchical clustering is proposed in [37], and the loss function of it includes both the overall loss of the entire hierarchy as well as the hierarchical information. He et al. [38] design the AGC module to adjust the graph structure and data features, and integrate AGC and AE with attention weights for heterogeneous features to learn more complex fused features. Peng et al. [39] propose a deep attention-guided graph clustering model with dual self-supervision, and the feature representation learned from it has multi-scale information from different layers. Recently, the deep clustering model on dynamic graph has also been proposed [40], which is closely related to the progress achieved in the work associated with dynamic graph representation learning [41,42]. Compared with static graph, dynamic graph contain richer information (e.g., the record of timestamps), but in general their models tend to have higher time and space complexity. Considering that the work of deep clustering on dynamic graph is in the initial stage and there are many commonalities between it and the work on deep clustering on static graph, here we still focus on the latter. Boosting the graph learning representation capability of the model certainly contributes to the improvement of clustering performance, there is also a benefit to providing better guidance for clustering task in the process of model learning feature representation. Inspired by [25], Xie et al. propose an objective function that is dependent on probability distributions in [43], which is different from the common clustering objective function represented by K-means and provides a better cluster assignment guidance for learned feature representation. On this basis, Xu et al. design a clustering objective function containing cluster-specificity distribution constraint in [44].

Nevertheless, even though the performance of these deep clustering models on graph has been partially improved, their access to guidance specific to the clustering task in the process of training is still inadequate, which makes the discrimination of feature representation they obtain for the clustering task remain unsatisfactory, and thus limit the potential of these models. Considering that the purpose pursued by the clustering task, the performance of these models may be further improved when we make the division between clusters more rational as well as more compact the intra-cluster distance of the original model output. Inspired by this, we intend to propose an optimization method for deep clustering models on graph. We expect our method to be like *mixup* [45], which is a data augmentation method for the image classification task in the field of computer vision that enhances the linear representation between training samples, and improves the classification accuracy of any such model.

3. Methodology

In this section, we introduce the construction of DCOM-GNN in detail. For a more intuitive description of DCOM-GNN, we show the relationship of the original deep clustering model on graph and the architecture after adding DCOM-GNN in Fig. 1. As we can see, in every epoch iteration, our DCOM-GNN can be understood as a "plug-and-play correction patch" to adjust the inter-cluster distance of the original model output in the last kth layer rationally, while making the intra-cluster distance of output more compact under the modulation of the original model output. Meanwhile, the architecture of DCOM-GNN is shown in Fig. 2, which consists of an inter-cluster distance optimization module and an intra-cluster distance optimization module.

3.1. Preliminaries

For each graph dataset, the nodes in it have a clear real meaning, so if we only focus on increasing the distance between clusters while ignoring the actual relationship between clusters in the process of inter-cluster distance optimization, it results in a decrease in the performance of deep clustering models on graph instead of an increase. Hence, we argue that to achieve our purpose, deep clustering must be based on rationality, which can be reflected by the similarity between clusters, i.e., the degree to which a cluster is influenced by other clusters should depend on the similarity between them.

In order to calculate the similarity, we need to acquire the feature representation and neighbor information of the cluster.

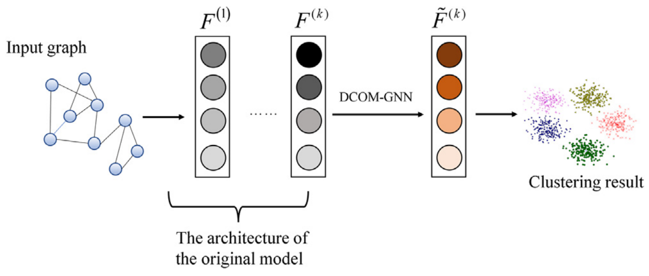


Fig. 1. The illustration of the relationship between the original deep clustering model on graph and DCOM-GNN. $F^{(1)}$ and $F^{(k)}$ are the feature representation learned from the *1*th layer and the last kth layer of the original model, respectively. $\tilde{F}^{(k)}$ is the new feature representation learned from the last kth layer after adding DCOM-GNN.

Assume that the original model output feature representation in the last kth layer of the deep clustering model on graph is $F^{(k)} \in R^{N\times D^{(k)}}$ (N and $D^{(k)}$ represent the number of nodes and dimension of the kth layer, respectively). Then we take $F^{(k)}$ as input, and assign nodes into clusters softly by a *sofatmax* function [19]. This process is shown by the first arrow in Fig. 2 and the formula for the cluster assignment matrix $\hat{F}^{(k)}$ is presented as follows:

$$\hat{F}^{(k)} = \text{soft}\max\left(F^{(k)}L^{(k)}\right) \tag{1}$$

where $L^{(k)} \in R^{D^{(k)}\times M}$ denotes a *linear* operation applied to the kth layer, which aims to change the dimensionality of $F^{(k)}$ to M (M denotes the number of clusters). The elements in $\hat{F}^{(k)}$ represent the probability that the nodes belong to each cluster.

3.2. Inter-cluster distance optimization module

For $\hat{F}^{(k)}$, where $\hat{F}^{(k)}[:, i] \in R^{N\times 1}$ represents the probabilities that all nodes belong to the ith cluster, which can also be interpreted as the feature representation of the ith cluster [46]. Similarly, $\hat{F}^{(k)}[j, :] \in R^{1\times M}$ denotes the feature representation of the jth node, from which we can also identify the probability that the node belongs to each cluster.

Until then, we have to discuss the two situations faced by the clustering task, i.e., non-overlapping and overlapping clustering task, and the difference between the two is whether each node belongs to only one cluster. When faced with the first type of task (i.e., each node belongs to only one cluster), we first adopt an operation *max-0* to process $\hat{F}^{(k)}$, and this process can be represented formally as follows:

$$\hat{F}^{(k)}_{\max-0}[j, :] = \max-0\left(\hat{F}^{(k)}[j, :]\right), j \in [1, \dots, N] \tag{2}$$

where the function of $\max-0\left(\hat{F}^{(k)}[j, :]\right)$ is to retain the maximum value in $\hat{F}^{(k)}[j, :]$ and set all other values to 0.

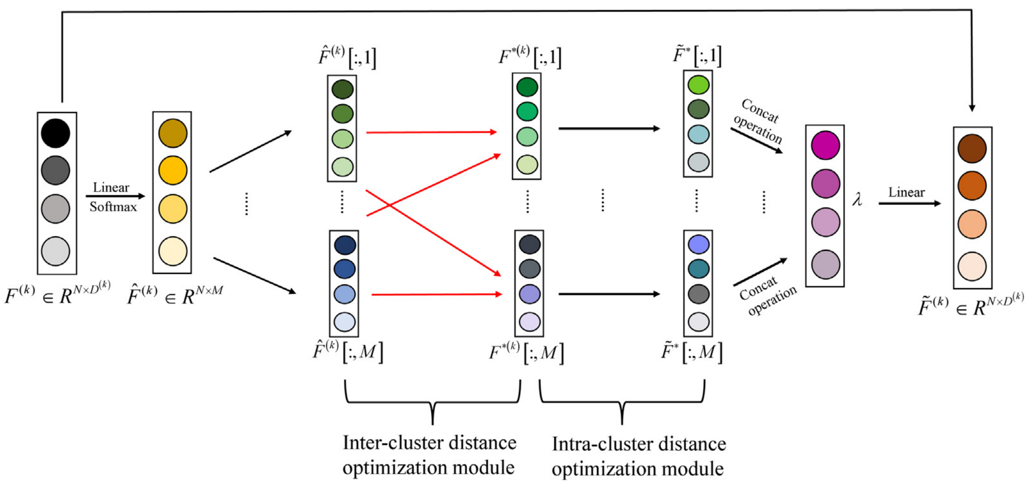
Furthermore, if we are faced with an overlapping clustering task, a soft processing scheme is adopted, and we process $\hat{F}^{(k)}$ in the following manner:

$$\hat{F}^{(k)}_{\text{top}-0}[j, :] = \text{top}-0\left(\hat{F}^{(k)}[j, :], \delta\right), j \in [1, \dots, N] \tag{3}$$

where δ is a hyperparameter, and the function of top-0 $\left(\hat{F}^{(k)}[j, :], \delta\right)$ is to change all except the top δ values in $\hat{F}^{(k)}[j, :]$ to 0.

Subsequently, according to $\hat{F}^{(k)}_{max-0}$ or $\hat{F}^{(k)}_{top-0}$, we can judge which clusters the model divides each node into and which nodes are contained in each cluster.

It is worth noting that in addition to the traditional similarity at the attribute level, we argue that the similarity between two



$F^{(k)} \in R^{N\times D^{(k)}}$
Linear
Softmax
$\hat{F}^{(k)} \in R^{N\times M}$
$\hat{F}^{(k)}[:,1]$
$\hat{F}^{(k)}[:,M]$
$F^{*(k)}[:,1]$
$F^{*(k)}[:,M]$
$\tilde{F}^{*}[:,1]$
$\tilde{F}^{*}[:,M]$
Concat operation
Concat operation
λ
Linear
$\tilde{F}^{(k)} \in R^{N\times D^{(k)}}$
Inter-cluster distance optimization module
Intra-cluster distance optimization module

Fig. 2. The red lines indicate that in the inter-cluster distance optimization phase, updating the feature representation of each cluster requires the participation from all clusters with different weight coefficients (as shown in Eq. (7)).

clusters also depends on the degree of connectivity between node pairs. Taking the citation network as an example, where nodes represent articles, connected node pairs indicate the existence of a citation relationship between them and the cluster is divided based on whether the article belongs to the same research field. Obviously, it is inconceivable that there is no citation between two close research fields. Meanwhile, it is found that more citation relationships exist between the two research fields meaning that they are more similar. Thus, to ensure the rationality of the inter-cluster distance adjustment, we also need to concentrate on the connected node pairs between two clusters besides the similarity between two clusters in vector space as a metric, and consider both the attribute and the structure perspective comprehensively.

Based on this idea, we propose the concept of adjacency confidence, which not only can reflect the connection situation of node pairs between clusters, but also plays the role of reducing the possible negative impact caused by the uncertainty of the model in the process of dividing the clusters. The formula for adjacency confidence degree $\left(\hat{F}^{(k)}_{\text{max-0}} \vee \hat{F}^{(k)}_{\text{top-0}}, A\right)_{ij}$ is expressed as follows:

$$\deg\text{ree}\left(\hat{F}^{(k)}_{\max-0} \vee \hat{F}^{(k)}_{\text{top-0}}, A\right)_{ij} = \sum_{\text{E}_{\text{ab}}}\left(\hat{F}^{(k)}_{\text{select}}[a,:] + \hat{F}^{(k)}_{\text{select}}[b,:]\right) \tag{4}$$

where the type of $\hat{F}^{(k)}_{\text{select}}$ depends on the clustering task faced with overlapping or non-overlapping, the former being $\hat{F}^{(k)}_{top-0}$ and the latter being $\hat{F}^{(k)}_{max-0}$. A is a known adjacency matrix, and we can obtain neighbor information of clusters by it and the location of the non-zero values in $\hat{F}^{(k)}_{top-0}$ or $\hat{F}^{(k)}_{max-0}$. E_{ab} denotes a set of node pairs connecting the ith cluster with the jth cluster.

Notably, the manner we design to obtain adjacency confidence degree $\left(\hat{F}^{(k)}_{\text{max-0}} \vee \hat{F}^{(k)}_{\text{top-0}}, A\right)_{ij}$ is implicit in distinguishing the importance of these node pairs based on the magnitude of the predicted probability value, and this idea of resorting to probability is common in deep clustering [47]. In brief, if there are connected node pairs between the ith cluster and the jth cluster, then the adjacency confidence degree $\left(\hat{F}^{(k)}_{\text{max-0}} \vee \hat{F}^{(k)}_{\text{top-0}}, A\right)_{ij}$ is the sum of the values in $\hat{F}^{(k)}_{max-0}$ or $\hat{F}^{(k)}_{top-0}$ for all the node pairs involved (e.g., assume that there is only one node pair between two clusters, then we sum the values in $\hat{F}^{(k)}_{max-0}$ or $\hat{F}^{(k)}_{top-0}$ corresponding to each of these two nodes), otherwise it is 0.

So far, we can acquire the weighting coefficient w_{ij} between the ith cluster and the jth cluster, the relevant formula is expressed as follows:

$$w_{ij} = \frac{\deg\text{ree}\left(\hat{F}^{(k)}_{\text{max-0}} \vee \hat{F}^{(k)}_{\text{top-0}}, A\right)_{ij}\tilde{w}_{ij}}{\sum_{p\in M}\deg\text{ree}\left(\hat{F}^{(k)}_{\text{max-0}} \vee \hat{F}^{(k)}_{\text{top-0}}, A\right)_{ip}\tilde{w}_{ip}} \tag{5}$$

$$\tilde{w}_{ij} = \frac{\text{sim}\left(\hat{F}^{(k)}[:,i], \hat{F}^{(k)}[:,j]\right)}{\sum_{p\in M}\text{sim}\left(\hat{F}^{(k)}[:,i], \hat{F}^{(k)}[:,p]\right)} \tag{6}$$

where $\text{sim}\left(\hat{F}^{(k)}[:,i], \hat{F}^{(k)}[:,j]\right)$ is the cosine similarity between the feature representation of the ith cluster and the jth cluster, and the larger the value means that the two are more similar.

Finally, the feature representation of the ith cluster after processing by our designed inter-cluster distance optimization module can be expressed as follows:

$$F^{*(k)}[:,i] = \sum_{j=1,j\neq i}^{M} w_{ij}\hat{F}^{(k)}[:,j] + \hat{F}^{(k)}[:,i] \tag{7}$$

3.3. Intra-cluster distance optimization module

After optimizing the inter-cluster distance for the output of the original deep clustering model on graph, then we concentrate on the intra-cluster distance, trying to make it more compact to further improve the clustering performance.

To achieve this goal, we expect all nodes in the same cluster to move towards one place according to some rule. Intuitively, this place requires the ability to reflect the centrality of a cluster. Inspired by K-means [4], we argue that the centroid of each cluster is a suitable choice. Thus, we average the feature representation of the nodes contained in the cluster for each dimension, and treat it as the centroid representation x_c (for ease of expression, here we denote the feature representation of node by x). For nodes that belong to the same cluster, we first regard x_c as the anchor node, then adjust the position of these nodes by measuring the current distance between their feature representation and x_c. Furthermore, due to the fact that the position of each node in the cluster is different, the "gravitation" they receive from the centroid should be distinct in the process of optimizing the intra-cluster distance, which are supposed to be proportional to the distance between the two, i.e., the farther the distance, the stronger the influence of the centroid on the node should be. In this way, the nodes belonging to the same cluster gather closer to the corresponding centroid. Meanwhile, for different clusters the number of nodes they contain varies, in general the cluster with a smaller number of nodes should occupy a smaller area than the cluster with a larger number of nodes, which means that the degree of influence of the centroid on the nodes is also affected by the number of nodes.

Thus, the intra-cluster distance optimization module is relevant not only to the distance between the centroid and the nodes, but also to the number of nodes contained in the cluster. After processing by this module, the feature representation of the ith node x_i in the cluster is updated to $\hat{x}_i$, and this process is expressed as follows:

$$\begin{aligned}\hat{x}_i &= \frac{\|x_i - x_c\|_2}{\sum_{j=1}^{p}\left\|x_j - x_c\right\|_2}x_i + \left[1 - \frac{\|x_i - x_c\|_2}{\sum_{j=1}^{p}\left\|x_j - x_c\right\|_2}\right]x_c \\ &= x_c + \frac{\|x_i - x_c\|_2}{\sum_{j=1}^{p}\left\|x_j - x_c\right\|_2}(x_i - x_c)\end{aligned} \tag{8}$$

where $\|x_i - x_c\|_2$ represents the distance between the ith node and the corresponding centroid. p is the number of nodes contained in the cluster, and the denominator part in Eq. (8) implies that the "gravitation" of the centroid on the nodes is also related to the number of nodes contained in the cluster.

Take the ith cluster as an example, when the feature representation of nodes in it is processed as above, we obtain the new feature representation of the ith cluster $\tilde{F}^{*(k)}[:, i]$.

3.4. Final output and algorithm

After the processing of the two modules mentioned above, the final output for the deep clustering model on graph after adding DCOM-GNN is expressed as follows:

$$\tilde{F}^{(k)} = F^{(k)} + \lambda\left[\text{Concat}\left(\tilde{F}^{*(k)}[:, 1]\cdots\tilde{F}^{*(k)}[:, M]\right)\bar{L}^{(k)}\right] \tag{9}$$

where $\bar{L}^{(k)} \in R^{M\times D^{(k)}}$ denotes a *linear* operation to recover the dimensionality of the feature representation after DCOM-GNN. λ is a hyperparameter, which is responsible for tuning the correction strength provided by the DCOM-GNN, and its range is 0 to 1.

In particular, since DCOM-GNN is a series of operations performed on the basis of the original model output, it does not act on the model from a global perspective as the objective function does, so if the final output depends only on the optimized result of DCOM-GNN, there is a possibility of corrupting the vector space due to over-correction, which will have a negative impact on the clustering performance of the model. Thus, to avoid over-correction, we resort to the original model output $F^{(k)}$ in Eq. (9), which can play a moderating role by providing a constraint to the DCOM-GNN to maintain the stability of the vector space. In other words, the purpose of adding DCOM-GNN to the deep clustering model on graph is to cooperate with the original model output and correct it to some extent, so as to achieve the goal of optimizing the model clustering performance, rather than to replace the original model output completely.

After the DCOM-GNN correction, the new output $\tilde{F}^{(k)}$ contains not only the features extracted from the original deep clustering models on graph, but also a "correction plugin" to adjust the inter-cluster distance of the original model output rationally, while enabling the intra-cluster distance to be more compact. Ideally, $\tilde{F}^{(k)}$ can adapt the clustering task better than $F^{(k)}$ learned from the original deep clustering model on graph. The pseudocodes of constructing DCOM-GNN are presented in Algorithm 1.

As shown in Fig. 2, the input $F^{(k)} \in R^{N\times D^{(k)}}$ is the feature representation learned from the original deep clustering model on graph. Then $\hat{F}^{(k)} \in R^{N\times M}$ is obtained from $F^{(k)}$ through a linear operation and *softmax* operation. The feature representation $\hat{F}^{(k)}[:, i] \in R^{N\times 1}$ of the ith cluster is updated to $F^{*(k)}[:, i] \in R^{N\times 1}$ after processing by inter-cluster distance optimization module. Afterwards, the feature representation of each node in the ith cluster is modified by the intra-cluster distance optimization module, and thus $F^{*(k)}[:, i]$ is updated to $\tilde{F}^{*(k)}[:, i] \in R^{N\times 1}$. Finally, the feature representation of each cluster is concatenated, followed by a linear operation, then multiplied by a hyperparameter λ and added with $F^{(k)}$ to obtain $\tilde{F}^{(k)} \in R^{N\times D^{(k)}}$.

Algorithm 1 A Deep Clustering Optimization Method for Graph Neural Networks

```
Input: The number of nodes $N$, the number of clusters $M$,
       the adjacency matrix $A$, the original model output
       in the last $k$-th layer $F^{(k)}$
Output: The new output in the last $k$-th layer $\tilde{F}^{(k)}$ after
        adding DCOM-GNN
 1: Transform $F^{(k)} \in R^{N\times D^{(k)}}$ into $\hat{F}^{(k)} \in R^{N\times M}$ via Eq.(1)
 2: if face a non-overlapping clustering task then
 3:     Process $\hat{F}^{(k)}$ via Eq.(2) to obtain $\hat{F}^{(k)}_{\text{max-0}}$
 4:     Obtain the neighbor information and adjacency
        confidence via $\hat{F}^{(k)}_{\text{max-0}}$ , $A$ and Eq.(4)
 5: end if
 6: if face an overlapping clustering task then
 7:     Process $\hat{F}^{(k)}$ via Eq.(3) to obtain $\hat{F}^{(k)}_{\text{top-0}}$
 8:     Obtain the neighbor information and adjacency
        confidence via $\hat{F}^{(k)}_{\text{top-0}}$ , $A$ and Eq.(4)
 9: end if
10: Calculate the weighting coefficients between clusters via
    Eqs.(5) and (6) based on the acquired neighbor information
    and adjacency confidence
11: Update the feature representation of the $i$-th cluster to $F^{*(k)}[:, i]$ via Eq.(7)
12: Update the node feature representation within the $i$-th cluster
    via Eq.(8) and obtain $\tilde{F}^{*(k)}[:, i]$
13: Calculate the final feature representation $\tilde{F}^{(k)}$ via Eq.(9)
14: return $\tilde{F}^{(k)}$
```

4. Experiments

In this section, we first introduce the datasets and metrics, followed by introducing referenced models used in the experiment. After that, we show the results of different experiments with discussion (here we use DCOM$_{\text{inter}}$ to denote the optimization method that only includes an inter-cluster distance optimization module, DCOM$_{\text{intra}}$ to denote the optimization method that only includes intra-cluster distance optimization module, and DCOM-GNN to denote the optimization method that includes both an inter-cluster distance optimization module and an intra-cluster distance optimization module).

4.1. Datasets

To ensure fairness, our experiments are completely based on the original deep clustering model on graph, and we use all the original papers' code from their github pages for each model. Meanwhile, due to the different datasets adopted by each author in the original papers, and taking into account the differences between the tasks these models are proficient in handling (e.g., models designed for the non-overlapping clustering task fail to handle the overlapping clustering task well). Thus, here we show the performance of the corresponding models before and

Table 1
The statistics of the datasets.

Dataset	#Nodes	#Features	#Classes
Facebook348	224	21	14
Facebook1684	786	15	17
Facebook1912	747	29	46
Engineering	14 927	4800	16
ACM	3025	1870	3
DBLP	4058	334	4
IMDB	3550	1007	3
Citeseer	3327	3703	6
Cora	2708	1433	7
Pubmed	19 717	500	3

after adding DCOM-GNN according to the type of clustering task, and this also enables it to recognize the differences that exist between these models (see Table 1).

For these datasets, nodes have a special meaning, and an edge between two nodes represents the existence of the relationship between them. The utilized datasets are described as follows:

Facebook. The original Facebook dataset is a social network dataset with more than one million node users, but for the purpose of protecting user privacy, the Facebook dataset as a public dataset is a small dataset divided by pre-processing on the original dataset. It includes Facebook348, Facebook414, etc., with the number of nodes ranging from 61 to 786, and the raw feature dimensions ranging from 6 to 21. These Facebook datasets are all for the overlapping clustering task.

Engineering. It is a co-authorship dataset for the overlapping clustering task. The node features in it are derived from the keywords of the papers by each author, and its raw feature dimension is 4800.

DBLP. DBLP as a classical citation dataset is applicable in many fields of graph research, which encompasses a total of four research directions in the field of computer research. The raw feature of each node in this dataset is a 2000-dimensional bag-of-words representation.

IMDB. It is a movie review dataset containing 3550 movie reviews, with node categories including action movies, comedies, and dramas. Among them, the node features can reflect the attitude of the reviewer towards the movie, and the raw feature of each node in this dataset is a 2000-dimensional bag-of-words representation are represented as a 1007-dimensional bag-of-words representation.

Citeseer. This dataset is a citation dataset, which contains 3327 articles in six research areas of machine learning, recording citation or cited information between papers. The raw feature of each node in this dataset is a 3703-dimensional bag-of-words representation.

ACM. It is a citation dataset, which contains a total of 3025 articles in three categories, the raw feature of each node in this dataset is an 1870-dimensional bag-of-words representation.

4.2. Referenced models and metrics

In order to enhance the persuasiveness and reflect the effectiveness of our proposed method, the reference models we employ are deep clustering models on graph that have been published in influential international conferences or journals in recent years, including AAAI, KDD, and WWW, etc.

NOCD [48]. The Neural Overlapping Community Detection (NOCD) is a model designed for the task of overlapping community detection that learns the affiliation matrix of the graph by the GCN model, and it uses maximum likelihood estimation to make the graph generated as similar as possible to the real graph, so as to obtain the cluster to which each node belongs.

SSGCAE [49]. To address the shortcomings of existing overlapping community detection methods in terms of combining link information with attribute information, a Semi-supervised Overlapping Community Detection Model with Graph Convolutional Autoencoder (SSGCAE) is proposed, which not only provides training in an end-to-end manner, but also integrates the prior information during this process.

HDMI [50]. High-order Deep Multiplex Infomax (HDMI) as an improvement of DEEP GRAPH INFOMAX (DGI) [21]. Compared with DGI, HDMI considers not only the external mutual information but also the internal mutual information, and HDMI designs a novel attention mechanism for nodes connected by multiple edges with different relationships in the network.

DMGI [51]. Consider a network in which nodes are connected by multiple types of relations. Unsupervised Attributed Multiplex Network Embedding (DMGI) is proposed to minimize the divergence between node embeddings of a specific relation type, integrating node embeddings of multiple graphs jointly in a systematic way.

SDCN [52]. To introduce structural information between data, Structural Deep Clustering Network (SDCN) computes a K-nearest neighbor graph based on the original data before initializing the model first, and uses it as the input to the GCN module. The entire model consists of roughly three parts: a deep neural network module, a graph convolution module, and a dual self-supervised module.

AGE [53]. Adaptive graph encoding method Adaptive Graph Encoder (AGE) is a novel framework for attribute graph embedding. It contains two advantages. One is AGE is the first proposal to use Laplace smoothing top-0 to mitigate the high-frequency noise signal from node features, the other is AGE uses an adaptive encoder to iteratively enhance the top-0ed features to better accomplish node embeddings.

DAEGC [47]. Deep Attentional Embedded Graph Clustering (DAEGC) extracts graph structure and attribute information in the encoding part utilizing a multi-layer encoder based on an attention mechanism. In the decoding part, DAEGC designs a self-training module, which guides the model to learn the feature representation for the clustering task by selecting some clustering soft labels with high confidence.

DFCN-RSP [54]. As a Deep Fusion Clustering Network with Reliable Structure Preservation (DFCN-RSP) aided by a random walk mechanism as well as a transformer-based graph autoencoder, it can capture more precise connection relations on the graph. On this basis, a dynamic cross-modality fusion strategy is also included in the model to obtain superior feature representation.

AGCC [55]. Since the presence of noise may cause the graph structure to turn unreliable, a model called Adaptive Graph Convolutional Clustering Network (AGCC) is proposed for this purpose. For AGCC, it can adjust the graph structure and node representation layer-by-layer with back-propagation to mitigate the negative effects due to inaccurate graph structure.

To be more convincing, we employ the most commonly clustering metric Normalized Mutual Information (NMI) [56], which is also one of the metrics in all the deep clustering models on graph experiments mentioned above. For NMI, a larger value implies a better clustering result.

We conducted our experiments on a machine with an NVIDIA Tesla V100 GPU (32 GB memory), 20-core Intel Xeon CPU (2.20 GHz), and 192 GB of RAM for each experiment.

4.3. Clustering results analysis

Tables 2 and 3 show the results of various models for the overlapping and non-overlapping clustering task in the original and with the addition of DCOM-GNN, respectively. Meanwhile, to

Table 2
Clustering results on models for the overlapping clustering task with $\text{DCOM}_{\text{inter}}$ & $\text{DCOM}_{\text{intra}}$ & DCOM-GNN added (The bold numbers represent the best results).

Model	Dataset			
	Facebook348	Facebook1684	Facebook1912	Engineering
NOCD	36.4	26.1	35.6	39.1
NOCD+$\text{DCOM}_{\text{inter}}$	43.0	31.4	38.2	42.4
NOCD+$\text{DCOM}_{\text{intra}}$	41.2	29.3	36.8	40.9
NOCD+DCOM-GNN	**43.7**	**32.3**	**38.8**	**43.6**
SSGCAE	37.5	36.0	31.3	43.2
SSGCAE+$\text{DCOM}_{\text{inter}}$	39.9	38.1	34.7	44.9
SSGCAE+$\text{DCOM}_{\text{intra}}$	39.6	37.5	34.5	45.8
SSGCAE+DCOM-GNN	**42.0**	**38.5**	**37.3**	**47.4**

Table 3
Clustering results on models for the non-overlapping clustering task with $\text{DCOM}_{\text{inter}}$ & $\text{DCOM}_{\text{intra}}$ & DCOM-GNN added (The bold numbers represent the best results).

Model	Dataset					
	IMDB	ACM	DBLP	Citeseer	Cora	Pubmed
HDMI	16.3	62.3	56.8	38.1	54.7	29.4
HDMI+$\text{DCOM}_{\text{inter}}$	18.8	63.9	58.4	38.3	56.5	30.6
HDMI+$\text{DCOM}_{\text{intra}}$	18.1	62.8	57.5	39.4	55.2	29.8
HDMI+DCOM-GNN	**19.4**	**65.0**	**60.4**	**40.4**	**57.0**	**31.2**
DMGI	19.6	70.2	55.4	37.9	55.1	28.8
DMGI+$\text{DCOM}_{\text{inter}}$	22.3	72.2	57.2	39.0	55.9	30.2
DMGI+$\text{DCOM}_{\text{intra}}$	21.9	71.3	55.9	38.4	55.5	30.0
DMGI+DCOM-GNN	**23.5**	**73.8**	**59.0**	**39.6**	**57.3**	**32.1**
SDCN	18.7	68.3	39.5	38.7	58.5	29.6
SDCN+$\text{DCOM}_{\text{inter}}$	22.4	69.1	41.9	39.9	59.1	32.0
SDCN+$\text{DCOM}_{\text{intra}}$	20.4	68.5	40.6	39.7	59.9	29.8
SDCN+DCOM-GNN	**23.3**	**70.9**	**42.6**	**40.5**	**60.2**	**32.3**
AGE	20.9	73.0	55.8	44.5	60.6	31.6
AGE+$\text{DCOM}_{\text{inter}}$	23.6	74.5	58.2	46.3	61.3	33.5
AGE+$\text{DCOM}_{\text{intra}}$	21.4	74.5	57.6	45.2	61.1	32.8
AGE+DCOM-GNN	**23.8**	**74.9**	**58.7**	**48.0**	**62.9**	**34.3**
DAEGC	17.7	65.4	55.7	39.7	52.8	26.6
DAEGC+$\text{DCOM}_{\text{inter}}$	18.5	68.9	57.4	41.4	53.1	28.7
DAEGC+$\text{DCOM}_{\text{intra}}$	19.2	66.3	56.8	41.3	54.8	28.4
DAEGC+DCOM-GNN	**19.9**	**70.1**	**57.6**	**42.1**	**55.6**	**30.9**
DFCN-RSP	21.5	74.1	51.2	46.3	61.9	32.9
DFCN+$\text{DCOM}_{\text{inter}}$	22.4	74.6	53.1	48.3	63.6	33.9
DFCN+$\text{DCOM}_{\text{intra}}$	21.9	74.7	52.8	46.5	62.3	33.6
DFCN+DCOM-GNN	**22.7**	**74.9**	**53.3**	**48.8**	**63.7**	**35.0**
AGCC	21.1	75.5	46.3	45.7	61.6	33.5
AGCC+$\text{DCOM}_{\text{inter}}$	23.9	76.0	47.2	46.8	62.3	34.8
AGCC+$\text{DCOM}_{\text{intra}}$	22.2	75.6	46.9	46.4	61.8	33.7
AGCC+DCOM-GNN	**24.2**	**76.3**	**48.6**	**47.2**	**62.4**	**35.5**

verify the effectiveness of the inter-cluster distance optimization module and the intra-cluster distance optimization module we proposed, we also add $\text{DCOM}_{\text{inter}}$ and $\text{DCOM}_{\text{intra}}$ individually to each model as two ablation experiments.

From Tables 2 and 3, we have the following observations:

The clustering performance for all models is improved by adding DCOM-GNN, which demonstrates the deep clustering model on graph benefits from adding DCOM-GNN. Specifically, these models learn better clustering feature representation after adding DCOM-GNN compared with their original case, and this is attributed to the capacity of DCOM-GNN to enhance the strength of the clustering-oriented guidance they receive.

The clustering performance after adding $\text{DCOM}_{\text{inter}}$ and $\text{DCOM}_{\text{intra}}$ individually is improved compared with the original model, which demonstrates the effectiveness of each of the two for optimizing the clustering performance of the model. Meanwhile, it confirms the model can be better adapted to the clustering task compared with the original model after adding these two modules.

Regardless of the model dealing with the overlapping clustering or non-overlapping clustering task, the clustering optimization effect of $\text{DCOM}_{\text{inter}}$ is better than that of $\text{DCOM}_{\text{intra}}$ in the majority of cases, which indicates that perhaps for most deep clustering models on graph, they more prone to improve the cohesiveness of clustering results while the effect of adjustment for inter-cluster distance is relatively limited.

For the original deep clustering models dealing with the overlapping clustering task, their performance varies for different datasets (e.g., NOCD is inferior to SSGCAE on Facebook1684, but outperforms SSGCAE on Facebook1912). The same phenomenon exists for the original deep clustering models dealing with the non-overlapping clustering task (e.g., DFCN is inferior to AGCC on ACM, but outperforms AGCC on DBLP)

Both for models dealing with the overlapping clustering task and non-overlapping clustering task, the optimization effect of adding DCOM-GNN depends on themselves. We argue that this is due to the unique structure of each model, and hence the benefits provided by DCOM-GNN differ for them.

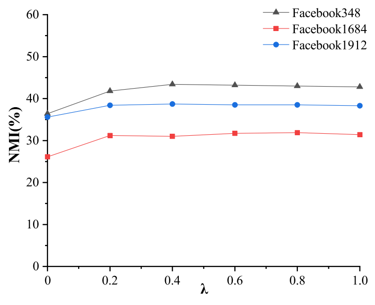
4.4. Hyperparameters analysis

As described in Eq. (9), λ is a critical hyperparameter used to balance the output of the original model and the output after adding DCOM-GNN. Therefore, we tend to explore the optimization effect of DCOM-GNN with $\lambda = \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ on various deep clustering models on graph (taking three datasets as an example), and it also is considered as the hyperparameter selection process of our experiment.

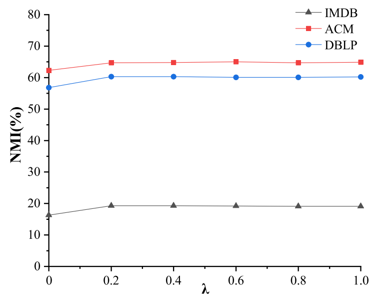
According to Fig. 3, we can see that when the value of λ is 0, the clustering performance of the model is inferior compared with the case when λ is other non-zero values, which confirms again that DCOM-GNN indeed contributes to the performance improvement of the deep clustering models on graph. Meanwhile, for these non-zero λ values, the difference in their corresponding clustering performance is not significant, we argue that this is due to the presence of the original model output $F^{(k)}$ in Eq. (9) acts as a moderator, which restricts the impact of DCOM-GNN on the model to a manageable range and avoiding over-correction.

In addition to λ, the presence of a hyperparameter δ in Eq. (3) is also a key hyperparameter. Thus, here we explore the effect of different values of δ on the optimization effect of the overlapping clustering task. Specifically, we select $\delta = \{1, 3, 5\}$ as the values and take NOCD as an example for the experiment on the three overlapping clustering datasets (i.e., Fackbook348, Fackbook1684 and Fackbook1912).

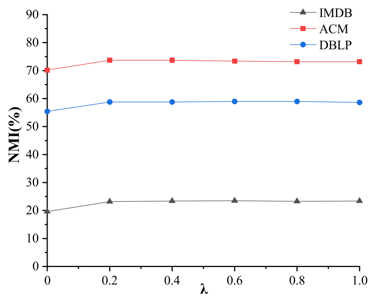
The results in Fig. 4 illustrate that for the overlapping clustering task, the choice of δ value has a certain impact on the final optimization effect. When the value of δ is 1, it is equivalent to treating the task as a non-overlapping clustering task, so the improvement in performance for the original model is lower compared with the other two values. Meanwhile, we suggest that a relatively large value of δ can be chosen first within a reasonable



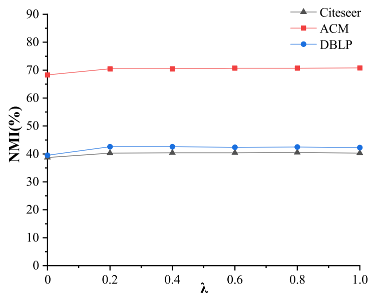
(a) NOCD



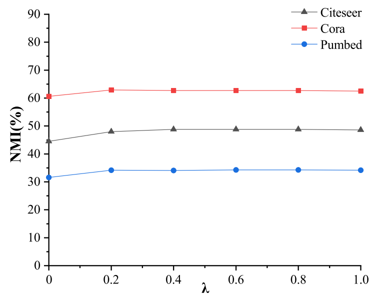
(b) HDMI



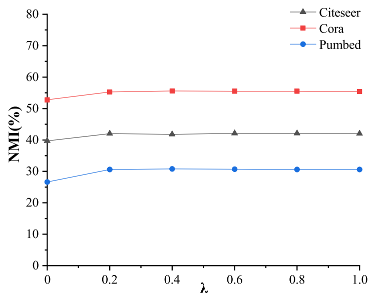
(c) DMGI



(d) SDCN

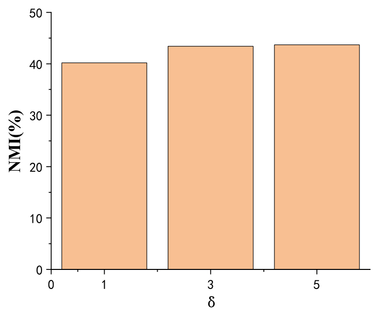


(e) AGE

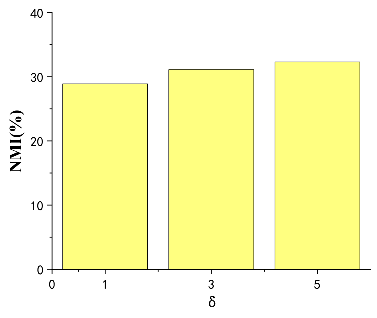


(f) DAEGC

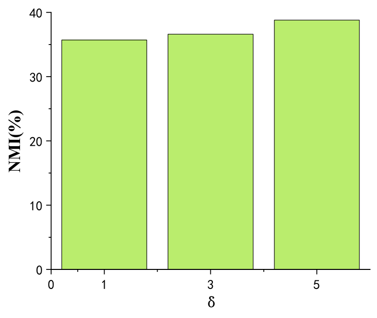
Fig. 3. Clustering results with different λ in DCOM-GNN for different deep clustering models on graph.



(a) Facebook348



(b) Facebook1684



(c) Facebook1912

Fig. 4. Clustering results after adding DCOM-GNN with different δ on NOCD for the overlapping clustering task.

range. The reason for this is that the adjacency confidence is obtained based on the predicted probability that nodes belong to each cluster in the original model. Theoretically, these probabilities differ significantly from each other, so even if the overlap degree of the dataset is not that high, the clusters with lower predicted affiliation probabilities have a limited impact on the final weighting coefficient.

4.5. Extended experiments

M-sensitivity Analysis: Furthermore, the number of clusters M is also important parameter for the clustering task. Although for deep clustering models on graph, the number of clusters into which the dataset has been divided is usually taken as the prior knowledge directly (we do the same thing in other experiments). In order to explore the effect of M on the results to further validate the effectiveness of DCOM-GNN under different settings of the original model, we take various values of $M = \{2, 4, 6, 8\}$ for six original deep clustering models on graph and each of them after adding DCOM-GNN, and conduct an experimental investigation.

From the results in Fig. 5, it can be observed that different Ms have a significant influence on the performance while DCOM-GNN improves the effect of the original deep clustering model on graph for any value of M. Thus, the results demonstrate again the superiority of DCOM-GNN in optimizing the clustering performance of these models and the fact that the optimization effect of DCOM-GNN varies depending on both the original model and dataset.

Training process analysis: To further validate the capability of DCOM-GNN in optimizing deep clustering models on graph, we treat AGE as the original model and compare its performance on different training epochs before and after the addition of DCOM-GNN. Meanwhile, we also show here the clustering performance when the final output is determined by DCOM-GNN only, i.e., the original output $F^{(k)}$ and hyperparameter λ are not included in Eq. (9) (denoted by DCOM_X), as an indication that the final output should be moderated by the original output.

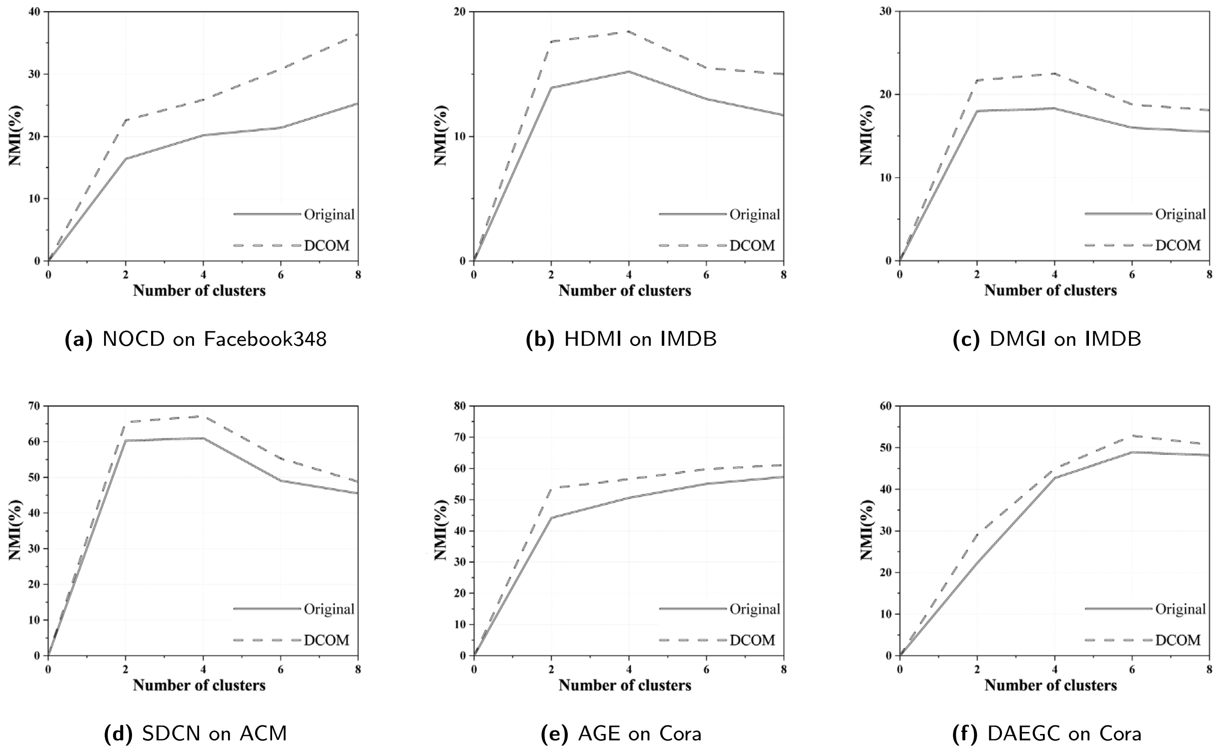


Fig. 5. Clustering results with different M on the original model and after adding DCOM-GNN.

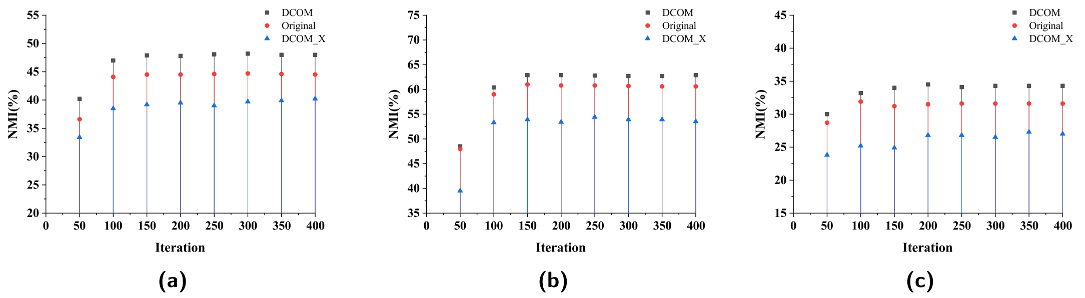


Fig. 6. Clustering results on Citeseer, Cora, and Pumbed dataset respectively with different training epochs for the original AGE model and after adding DCOM-GNN.

We can observe from the results in Fig. 6 that the entire training process of the model benefits from the addition of DCOM-GNN, while not hindering the convergence of the model, which also means that DCOM-GNN does not overly aggravate the computational burden of the model. In addition, according to the results of DCOM_X, we can find that when the final output is determined only by the output of DCOM-GNN, the clustering performance of the model is degraded, which suggests that DCOM-GNN indeed leads to over-correction, and reflects the necessity of the presence of the original output that plays a moderating role in Eq. (9).

Boundary Nodes Analysis: Boundary nodes refer to the nodes in a cluster that are relatively far from the centroid, and the reason we discuss these nodes specifically is that intuitively these nodes may imply that the original deep clustering model on graph has difficulty in dividing them, i.e., these nodes belong to that cluster with less confidence. Thus, especially when these nodes are processed by the intra-cluster distance optimization module may have a negative impact on the clustering performance of the model, which can be considered as a further exploration of over-correction.

We take SDCN as the original deep clustering model on graph, and first process its original clustering results on the three datasets Citeseer, ACM, and DBLP. Specifically, we set a ratio to fix some nodes so that these nodes are not affected by the intra-cluster distance optimization module after the DCOM-GNN is added, keeping their distance to the centroid in their respective clusters constant. It is noted that to reflect the concept of boundary nodes, the selection of these fixed nodes is determined by their distance from the centroid, i.e., we fix a certain ratio of nodes in each cluster that is farthest from the centroid. We show in Table 4 the clustering results after adding DCOM-GNN on SDCN with different fixed ratios.

Table 4
Clustering results (in NMI%) after adding DCOM-GNN on SDCN with different fixed ratios.

Ratio	Dataset		
	Citeseer	ACM	DBLP
0%	**40.5**	**70.9**	**42.6**
10%	40.1	69.8	42.3
20%	39.9	69.4	42.0
30%	39.7	69.3	41.8

According to Table 4, we can observe that the optimization effectiveness of DCOM-GNN for the model gradually decreases

Table 5
The time difference for deep clustering models on graph before and after the adding DCOM-GNN.

	Model					
	NOCD +DCOM-GNN	HDMI +DCOM-GNN	DMGI +DCOM-GNN	SDCN +DCOM-GNN	AGE +DCOM-GNN	DAEGC +DCOM-GNN
Dataset	Facebook348	DBLP	DBLP	DBLP	Cora	Cora
Time difference (in s)	0.02	0.09	0.10	0.09	0.07	0.06

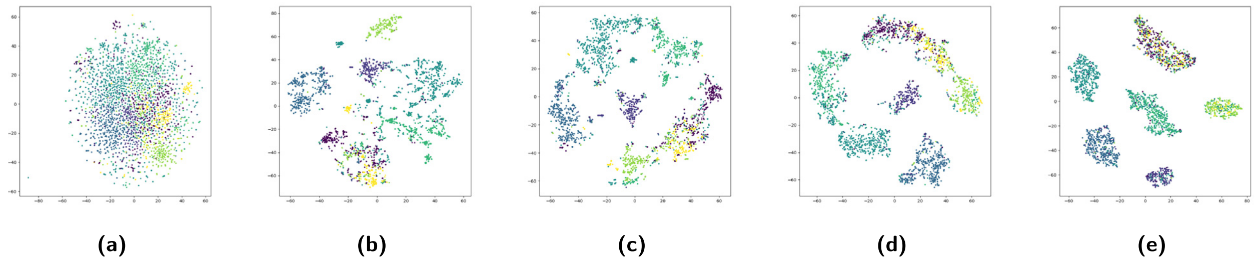


Fig. 7. Visualization results on Cora dataset. (a), (b), (c), (d), and (e) represent the raw feature representation, the feature representation learned from the original DAEGC model, after adding only the inter-cluster distance optimization module, after adding only the intra-cluster distance optimization module, after adding DCOM-GNN, respectively.

as the fixed ratio increases. We argue that the reason for this is first since our DCOM-GNN can enhance the strength of the clustering-oriented guidance received by the model, so fixing some nodes in the process is equivalent to limiting the potential of the DCOM-GNN to optimize the original model output less strongly. Secondly, we have confirmed in the previous experiment that the original model output can effectively avoid over-correction, so it is not necessary to perform special operations for some nodes, which will instead prevent these nodes from benefiting from DCOM-GNN.

Time Cost Analysis: As an optimization method, DCOM-GNN requires attachment to a deep clustering model on graph to work, so the training time cost of the model before and after the addition of DCOM-GNN is also a concern. Thus, we perform relevant experiments based on six models and show the results in Table 4. Notably, considering that each model has a different architectural complexity and their default number of training iterations is not the same, it is clearly unfair to compare their total training time before and after adding DCOM-GNN. Hence, the value we present in Table 5 is the difference in time between each training epoch before and after adding DCOM-GNN for these models on the corresponding dataset.

As we can observe in Table 5, adding DCOM-GNN does not significantly increase the training time cost of the model, which is attributed to we implementation of DCOM-GNN in the pytorch framework employing matrix operations and a set of default functions, and this has an extremely limited impact on the efficiency of pytorch computing. Moreover, this confirms that the increase in time cost after adding DCOM-GNN is mainly related to the properties of the dataset itself.

Visualization Analysis: To display the superiority of our proposed DCOM-GNN for improving the performance of the deep clustering model on graph, we conduct a visualization experiment with the Cora dataset and the DAEGC-based deep clustering model as an example, which shows the discrimination of the feature representation learned from the different cases for the clustering task.

According to the visualization results in Fig. 7, we can draw some interesting observations. First, both the inter-cluster optimization module and the intra-cluster optimization module indeed contribute to the original deep clustering model to obtain the higher quality feature representation for the clustering task. Second, when adding two optimization modules individually, both of them optimize another distance to some extent. Finally, compared with adding the inter-cluster distance optimization module or intra-cluster distance optimization module separately, adding DCOM-GNN can provide more adequate clustering-oriented guidance to the original deep clustering model, so its learned feature representation is more discriminative for the clustering task.

5. Conclusion

In this paper, considering that the feature representation learned from most existing deep clustering models on graph is not sufficiently discriminative for the clustering task, resulting in models that only yield sub-optimal clustering results. Therefore, we propose a Deep Clustering Optimization Method for Graph Neural Networks (DCOM-GNN) to enable these models to learn better feature representation for the clustering task. As a "plug-and-play correction patch" that can be easily attached to the original model, DCOM-GNN serves not only to adjust the inter-cluster distance of the original model output rationally, but also to make its intra-cluster distance more compact through an inter-cluster distance optimization module and an intra-cluster distance optimization module. Experiments demonstrate that the addition of DCOM-GNN can significantly improve the performance of various deep clustering models on graph. In future work, we will further explore combining DCOM-GNN with more deep clustering models on graph, as a way to further validate the effectiveness of DCOM-GNN for improving the clustering performance of such models. Meanwhile, we will also endeavor to reduce the dependence of the final output from DCOM-GNN on the original model output.

CRediT authorship contribution statement

Haoran Yang: Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization. **Junli Wang:** Writing – original draft, Data curation. **Rui Duan:** Visualization, Supervision. **Chungang Yan:** Supervision, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request

Acknowledgments

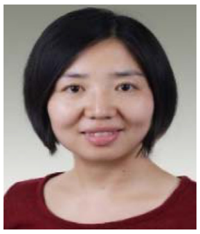
This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB2100801 and in part by the Shanghai Science and Technology Commission Project under Grant 19511101300.

References

[1] S. Yang, S. Verma, B. Cai, J. Jiang, K. Yu, F. Chen, S. Yu, Variational co-embedding learning for attributed network clustering, Knowl.-Based Syst. 270 (2023) 110530.
[2] Q. Wang, Z. Tao, W. Xia, Q. Gao, X. Cao, L. Jiao, Adversarial multiview clustering networks with adaptive fusion, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[3] Y. Ding, Z. Zhang, X. Zhao, Y. Cai, S. Li, B. Deng, W. Cai, Self-supervised locality preserving low-pass graph convolutional embedding for large-scale hyperspectral image clustering, IEEE Trans. Geosci. Remote Sens. 60 (2022) 1–16.
[4] K. Krishna, M.N. Murty, Genetic K-means algorithm, IEEE Trans. Syst. Man Cybern. B 29 (3) (1999) 433–439.
[5] U. Von Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (4) (2007) 395–416.
[6] B. Yang, X. Fu, N.D. Sidiropoulos, M. Hong, Towards k-means-friendly spaces: Simultaneous deep learning and clustering, in: International Conference on Machine Learning, PMLR, 2017, pp. 3861–3870.
[7] P. Ji, T. Zhang, H. Li, M. Salzmann, I. Reid, Deep subspace clustering networks, in: Advances in Neural Information Processing Systems, Vol.30, 2017.
[8] X. Yang, C. Deng, F. Zheng, J. Yan, W. Liu, Deep spectral clustering using dual autoencoder network, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4066–4075.
[9] S. Min, Z. Gao, J. Peng, L. Wang, K. Qin, B. Fang, STGSN—A spatial–temporal graph neural network framework for time-evolving social networks, Knowl.-Based Syst. 214 (2021) 106746.
[10] A. Salamat, X. Luo, A. Jafari, HeteroGraphRec: A heterogeneous graph-based neural networks for social recommendations, Knowl.-Based Syst. 217 (2021) 106817.
[11] J. Gao, J. Wu, X. Zhang, Y. Li, C. Han, C. Guo, Partition and learned clustering with joined-training: Active learning of GNNs on large-scale graph, Knowl.-Based Syst. 258 (2022) 110050.
[12] R. Duan, C. Yan, J. Wang, C. Jiang, Class-homophilic-based data augmentation for improving graph neural networks, Knowl.-Based Syst. 269 (2023) 110518.
[13] M. Guang, C. Yan, Y. Xu, J. Wang, C. Jiang, A multichannel convolutional decoding network for graph classification, IEEE Trans. Neural Netw. Learn. Syst. (2023).
[14] Y. Xu, J. Wang, M. Guang, C. Yan, C. Jiang, Multistructure graph classification method with attention-based pooling, IEEE Trans. Comput. Soc. Syst. (2022).
[15] D. Cheng, X. Wang, Y. Zhang, L. Zhang, Graph neural network for fraud detection via spatial-temporal attention, IEEE Trans. Knowl. Data Eng. 34 (8) (2020) 3800–3813.
[16] F. Tian, B. Gao, Q. Cui, E. Chen, T.-Y. Liu, Learning deep representations for graph clustering, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 28, 2014.
[17] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1225–1234.
[18] T. Ke, C. Peng, W. Xiao, P.S. Yu, W. Zhu, Deep recursive network embedding with regular equivalence, in: The 24th ACM SIGKDD International Conference, 2018.
[19] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations (ICLR), 2017, pp. 1–14.
[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.
[21] P. Velickovic, W. Fedus, W.L. Hamilton, P. Liò, Y. Bengio, R.D. Hjelm, Deep graph infomax, ICLR (Poster) 2 (3) (2019) 4.
[22] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, C.-J. Hsieh, Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 257–266.
[23] Z. Peng, H. Liu, Y. Jia, J. Hou, Attention-driven graph clustering network, in: Proceedings of the 29th ACM International Conference on Multimedia, 2021, pp. 935–943.
[24] X. Zhang, H. Liu, Q. Li, X.M. Wu, Attributed graph clustering via adaptive graph convolution, in: 28th International Joint Conference on Artificial Intelligence, IJCAI 2019, 2019, pp. 4327–4333, International Joint Conferences on Artificial Intelligence.
[25] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: International Conference on Machine Learning, PMLR, 2016, pp. 478–487.
[26] L. Van der Maaten, G. Hinton, Visualizing data using t-sne, J. Mach. Learn. Res. 9 (11) (2008).
[27] X. Guo, X. Liu, E. Zhu, J. Yin, Deep clustering with convolutional autoencoders, in: International Conference on Neural Information Processing, Springer, 2017, pp. 373–382.
[28] X. Guo, L. Gao, X. Liu, J. Yin, Improved deep embedded clustering with local structure preservation, in: Ijcai, 2017, pp. 1753–1759.
[29] T.-E. Lin, H. Xu, H. Zhang, Discovering new intents via constrained deep adaptive clustering with cluster refinement, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 8360–8367.
[30] X. Guo, E. Zhu, X. Liu, J. Yin, Deep embedded clustering with data augmentation, in: Asian Conference on Machine Learning, PMLR, 2018, pp. 550–565.
[31] N. Mrabah, M. Bouguessa, R. Ksantini, Adversarial deep embedded clustering: on a better trade-off between feature randomness and feature drift, IEEE Trans. Knowl. Data Eng. (2020).
[32] T.N. Kipf, M. Welling, Variational graph auto-encoders, in: Neural Information Processing Systems, 2016, pp. 722–730.
[33] P. Hu, K.C. Chan, T. He, Deep graph clustering in social network, in: Proceedings of the 26th International Conference on World Wide Web Companion, 2017, pp. 1425–1426.
[34] S. Pan, R. Hu, S.-f. Fung, G. Long, J. Jiang, C. Zhang, Learning graph embedding with adversarial training methods, IEEE Trans. Cybern. 50 (6) (2019) 2475–2487.
[35] H. Zhang, P. Li, R. Zhang, X. Li, Embedding graph auto-encoder for graph clustering, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[36] J. Cheng, Q. Wang, Z. Tao, D. Xie, Q. Gao, Multi-view attribute graph convolution networks for clustering, in: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, 2021, pp. 2973–2979.
[37] Y. Xing, T. He, T. Xiao, Y. Wang, Y. Xiong, W. Xia, D. Wipf, Z. Zhang, S. Soatto, Learning hierarchical graph neural networks for image clustering, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 3467–3477.
[38] X. He, B. Wang, Y. Hu, J. Gao, Y. Sun, B. Yin, Parallelly adaptive graph convolutional clustering model, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[39] Z. Peng, H. Liu, Y. Jia, J. Hou, Deep attention-guided graph clustering with dual self-supervision, 2021, arXiv preprint arXiv:2111.05548.
[40] M. Liu, Y. Liu, K. Liang, S. Wang, S. Zhou, X. Liu, Deep temporal graph clustering, 2023, arXiv preprint arXiv:2305.10738.
[41] Y. Jia, Z. Gu, Z. Jiang, C. Gao, J. Yang, Persistent graph stream summarization for real-time graph analytics, World Wide Web (2023) 1–21.
[42] M. Liu, J. Wu, Y. Liu, Embedding global and local influences for dynamic graphs, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 4249–4253.
[43] L. Guo, Q. Dai, Graph clustering via variational graph embedding, Pattern Recognit. 122 (2022) 108334.
[44] H. Xu, W. Xia, Q. Gao, J. Han, X. Gao, Graph embedding clustering: Graph attention auto-encoder with cluster-specificity distribution, Neural Netw. 142 (2021) 221–230.
[45] H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, Mixup: Beyond empirical risk minimization, in: International Conference on Learning Representations, 2018.
[46] K. Zhou, X. Huang, Y. Li, D. Zha, R. Chen, X. Hu, Towards deeper graph neural networks with differentiable group normalization, Adv. Neural Inf. Process. Syst. 33 (2020) 4917–4928.
[47] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, C. Zhang, Attributed graph clustering: a deep attentional embedding approach, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, 2019, pp. 3670–3676.

[48] O. Shchur, S. Günnemann, Overlapping community detection with graph neural networks, in: Deep Learning on Graphs Workshop, KDD, 2019.
[49] C. He, Y. Zheng, J. Cheng, Y. Tang, G. Chen, H. Liu, Semi-supervised overlapping community detection in attributed graph with graph convolutional autoencoder, Inform. Sci. 608 (2022) 1464–1479.
[50] B. Jing, C. Park, H. Tong, Hdmi: High-order deep multiplex infomax, in: Proceedings of the Web Conference 2021, 2021, pp. 2414–2424.
[51] C. Park, D. Kim, J. Han, H. Yu, Unsupervised attributed multiplex network embedding, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 5371–5378.
[52] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, P. Cui, Structural deep clustering network, in: Proceedings of the Web Conference 2020, 2020, pp. 1400–1410.
[53] G. Cui, J. Zhou, C. Yang, Z. Liu, Adaptive graph encoder for attributed graph embedding, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 976–985.
[54] L. Gong, W. Tu, S. Zhou, L. Zhao, Z. Liu, X. Liu, Deep fusion clustering network with reliable structure preservation, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[55] X. He, B. Wang, R. Li, J. Gao, Y. Hu, G. Huo, B. Yin, Graph structure learning layer and its graph convolution clustering application, Neural Netw. (2023).
[56] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, P.S. Yu, Heterogeneous graph attention network, in: The World Wide Web Conference, 2019, pp. 2022–2032.

Haoran Yang, received the M.S. degree s from the Anhui University of Science and Technology, Huainan, China, in 2020. He is currently pursuing the Ph.D. degree with the Department of Computer Science, Tongji University, Shanghai, China. His current research interests include graph neural networks and graph representation learning.


Junli Wang, received the Ph.D. degree in computer science from Tongji University, Shanghai, China, in 2007. She is currently an Associate Researcher with the College of Electronics and Information Engineering, Tongji University. Her research interests include text data analysis, deep learning, and artificial intelligence.

Rui Duan, received his bachelor's and master's degrees from the Anhui University of Science and Technology in 2016 and 2019, Huainan, China. He is currently pursuing the Ph.D. degree with the Department of Computer Science, Tongji University, Shanghai, China. His main research directions include Graph Neural Networks, Petri nets theory and its application.

Chungang Yan, received the Ph.D. degree from Tongji University, Shanghai, China, in 2006. She is currently a Professor with the Department of Computer Science and Technology, Tongji University. She has published more than 100 papers in domestic and international academic journals and conference proceedings. Her current research interests include concurrent model and algorithm, Petri net theory, formal verification of software, and trusty theory on software process.