Knowledge-Based Systems 317 (2025) 113322

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Sharpening deep graph clustering via diverse bellwethers

Peiyao Zhao [a], Xin Li [a,*], Yuangang Pan [b,*], Ivor W. Tsang [b], Mingzhong Wang [c], Lejian Liao [a]

[a] Beijing Institute of Technology, No. 5, Zhongguancun South Street, Haidian District, Beijing, 100081, China
[b] Agency for Science, Technology and Research (A*STAR), Singapore
[c] University of the Sunshine Coast, Sunshine Coast, Australia

ARTICLE INFO

Keywords:
Deep graph clustering
Diverse cluster centroids
Sharpening clustering assignment
Discriminative node representations

ABSTRACT

Deep graph clustering has attracted increasing attention in data analysis recently, which leverages the topology structure and attributes of graph to divide nodes into different groups. Most existing deep graph clustering models, however, have compromised performance due to a lack of discriminative representation learning and adequate support for learning diverse clusters. To address these issues, we proposed a Diversity-promoting Deep Graph Clustering (DDGC) model that attains the two essential clustering principles of minimizing the intra-cluster variance while maximizing the inter-cluster variance. Specifically, DDGC iteratively optimizes the node representations and cluster centroids. First, DDGC maximizes the log-likelihood of node representations to obtain cluster centroids, which are subjected to a differentiable diversity regularization term to force the separation among clusters and thus increase inter-cluster variances. Moreover, a minimum entropy-based clustering loss is proposed to sharpen the clustering assignment distributions in order to produce compact clusters, thereby reducing intra-cluster variances. Extensive experimental results demonstrate that DDGC achieves state-of-the-art clustering performance and verifies the effectiveness of each component on common real-world datasets. Experiments also verify that DDGC can learn discriminative node representations and alleviate the *over-smoothing* issue.

1. Introduction

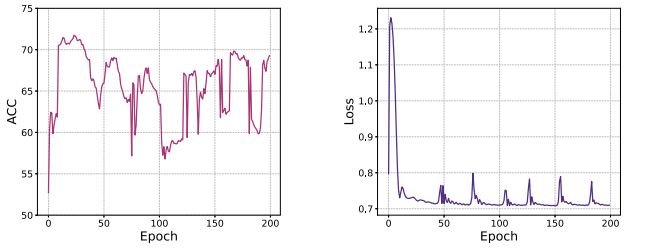
Graph clustering, a fundamental technique in data analysis, aims to partition all nodes within a graph into k distinct clusters. Ideally, nodes within the same cluster have more similar topological features and attribute values than nodes in other clusters. Typical applications of graph clustering include community detection [1–3], group segmentation [4–6], image segmentation [7–10], domain adaptation [11–13]. The conventional graph clustering algorithms [14,15] have demonstrated their effectiveness in grouping nodes based on well-defined handcrafted features and well-established similarities. However, their effectiveness heavily relies on the feature selection and the initial centroid assignment. Recently, "deep graph clustering" has emerged as a promising approach by incorporates deep learning frameworks, such as Deep Neural Networks (DNN) [16], Graph Neural Networks (GNN) [5,17,18], with the conventional clustering algorithms. This fusion aims to transform graph data from an original complex structure space to a low-dimensional feature space to effectively facilitate the task.

Most existing deep graph clustering algorithms are *Two-step Methods* [15,19,20], including (1) first pretrain a graph encoder to obtain low-dimensional node representations via minimizing an unsupervised or self-supervised graph representation learning loss, such as the reconstruction loss or contrastive learning loss [21], and then (2) performing conventional clustering algorithms like k-means [15,22] or spectral clustering [14,23] on the learned node representations to obtain the clustering results. Although these algorithms have achieved attractive clustering performance, an emerging challenge lies in the fact that the node representations learned by the first step are not dedicatedly designed or tailored for clustering tasks in the second step, leading to sub-optimal clustering results.

To bridge these two steps, after pretraining an encoder and cluster centroids like *Two-step methods*, DAEGC [24] first introduces a fine-tuning stage to simultaneously training node representations and centroids by minimizing a Kullback–Leibler (KL) clustering loss. Recent studies mainly focus on designing clustering loss without relying any labels [25] and resorting to advanced techniques to develop effective

* Corresponding authors.
E-mail addresses: peiyaozhao@bit.edu.cn (P. Zhao), xinli@bit.edu.cn (X. Li), pan_yuangang@cfar.a-star.edu.sg (Y. Pan), ivor_tsang@cfar.a-star.edu.sg (I.W. Tsang), mwang@usc.edu.au (M. Wang), liaolj@bit.edu.cn (L. Liao).

https://doi.org/10.1016/j.knosys.2025.113322
Received 27 April 2024; Received in revised form 17 October 2024; Accepted 7 March 2025
Available online 31 March 2025



(a) Clustering ACC (%) of GAE (b) Reconstruction Loss of GAE

Fig. 1. The clustering accuracy (ACC) and reconstruction loss of GAE in the pretraining process on the Cora dataset. With the convergence of the loss curve, the accuracy of GAE exhibits persistent oscillations, failing to converge, thus struggling to yield clustering-friendly node representations.

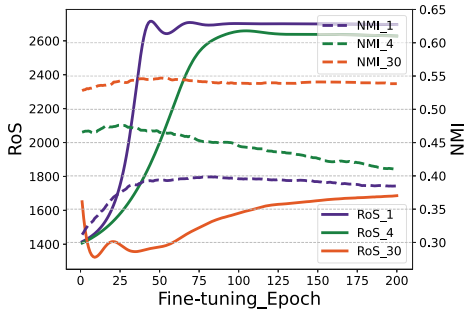


Fig. 2. The NMI and RoS matrices of the fine-tuning method, DAEGC, on the Cora dataset. Initialized with the pre-trained encoder at 1, 4, and 30 epochs in Fig. 1, DAEGC yields the clustering results represented by purple, green, and orange colors. Notably, NMI and RoS exhibit a negative correlation. Higher NMI values and lower RoS values indicate better clustering results.

graph encoder [26,27], which can enable models capture more semantic information for benefiting representation learning to facilitate better clusters. However, all those methods typically treat centroids as parameters and fine-tune them with graph encoder via minimizing a KL-based clustering loss in the fine-tuning stage. We refer to this type of methods as *Fine-tuning Methods*. The detailed comparison is shown in Fig. 4. In fact, both *Two-step Methods* and *Fine-tuning Methods* typically rely heavily on an well-pretrained graph encoder to start up the graph clustering task.

Although *Fine-tuning Methods* have achieved promising results, our analysis still reveal significant limitations of the widely-used graph encoder, GAE, and the associated KL-divergence clustering loss in the *Fine-tuning Methods*. Specifically, even a well-pretrained encoder, *e.g.*, GAE, fails to yield clustering-friendly node representations. As depicted in Fig. 1, the accuracy (ACC) exhibits extreme fluctuation throughout pretraining epochs, despite the early convergence of the reconstruction loss of GAE. This fluctuation arises because the neighbor aggregation mechanism in graph convolution of GAE tend to overly smooth the node representations, thereby reducing their discrimination. Consequently, nodes struggle to confidently find the appropriate group, significantly impacting the clustering performance. So strengthening node discrimination is important to improve the graph clustering performance.

Moreover, the limitation of high dependency on the quality of the pretrained encoder is demonstrated in Fig. 2. Take DAEGC [24] as an example of the *Fine-tuning Methods*, we use pretrained encoders of different epochs as the initializations in the fine-tuning stage, including encoders at 1, 4, and 30 pretraining epochs from Fig. 1, corresponding to the least effective, moderately effective, and most effective pretrained encoders. After fine-tuning, in Fig. 2, the final clustering results yielded by DAEGC, labeled as NMI_1, NMI_4, and NMI_30, clearly demonstrate the lowest, better, and best clustering performance, which closely mirrors the quality of the pretrained graph encoder. This phenomenon verifies that *Fine-tuning Methods* are very sensitive to the pretraining encoder. That is because applying the conventional KL-divergence as the clustering loss can prematurely restrict the exploration of representation space. Specifically, in KL-based clustering loss, the cluster centers and encoder are coupled as parameters, both need to be optimized together through gradient back-propagation. This highlights the importance of encouraging GAE to be pretrained solely in the pretraining stage of *Fine-tuning Methods* .

To examine how *Fine-tuning Methods* are influenced by the pretrained encoder, we introduce a metric, Ratio of Similarity (RoS), which quantifies the distinctiveness of node representations in the context of clustering:

$$\text{RoS} = \frac{n^2 \sum_{i,j=1}^{m} \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j}{m^2 \sum_{i,j=1}^{n} \mathbf{z}_i^\top \mathbf{z}_j}, \tag{1}$$

where $\{\boldsymbol{\mu}_i\}_{i=1}^{m}$ denotes m cluster centroids, and $\{\mathbf{z}_i\}_{i=1}^{n}$ denotes n node representations. A lower RoS indicates a larger degree of separation between clusters. We observe the change of ROS with the increase of fine-tuning epoch. As illustrated in Fig. 2, the fine-tuning method, DAEGC, initialized using the best/worst pretrained encoder at 1/30 pretraining epochs, performs the highest/lowest Normalized Mutual Information (NMI) (orange/purple dashed line), but yields the lowest/highest RoS values (orange/purple solid line), respectively. This implies that RoS is generally negatively correlated to clustering performance; the lower the RoS, the higher the NMI. This finding motivates us to diversify centroids and separate clusters to enhance node discrimination for improving the clustering task.

In this paper, we propose a novel Diversity-promoting Deep Graph Clustering model (DDGC) that simultaneously pursues two essential principles of maximizing inter-cluster variance and minimizing intra-cluster variances, as illustrated in Fig. 3. Specifically, DDGC iteratively updates the cluster centroids and node representations rather than coupling them to update in the KL-divergence clustering loss. We maximize the log-likelihood function of node representations to learn cluster centroids, where a diversity regularization term among centroids is exerted to gradient for diversifying centroids, thus increasing the inter-cluster variances. Besides, we propose a Minimum Entropy (ME)-based clustering loss to sharpen the assignment distribution for reducing the intra-variances, which makes centroids guide intra-cluster nodes to move toward their positions and directions. We refer centroids as "bellwethers", due to the leading update effect of latent node features within clusters. The cohorts of nodes follow the diverse bellwethers, to update. By seeking diverse bellwethers, different clusters become more separated. Finally, we can obtain well-separate clusters with wider cluster boundaries. Fig. 13(b) in Appendix A illustrates the dynamic process of our "bellwethers" leading representation learning. Besides, a gradient analysis between the conventional KL- and proposed ME-based clustering loss is provided to verify the advantage of the mechanism of sharpening the assignment distribution in our ME-based clustering loss. Our DDGC model achieves state-of-the-art clustering performance by implementing techniques of diversifying centroids and sharpening assignment distribution. These techniques contribute to stabilizing DDGC trained from scratch. Additionally, DDGC is verified to learn discriminative representations, particularly in alleviating *over-smoothing*. The main contributions of this paper are summarized as follows:

- **A new graph clustering model.** We propose a novel graph clustering model, DDGC, which iteratively updates the cluster centroids and node representations. The proposed seeking-diverse-bellwethers module is designed to learn diverse cluster centroids, and then our ME-based clustering loss and reconstruction loss are

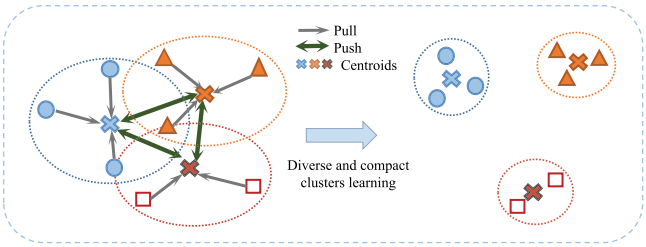


Fig. 3. Pulling nodes toward their centroid and pushing centroids away from each other can minimize the intra-cluster variance while maximizing the inter-cluster variance, leading to diverse and compact clusters.

minimized together to update node representations. The sharpening assignment mechanism will force nodes in groups to keep up with the bellwethers to produce compact clusters.

- **Diversified centroids.** We propose to maximize the log-likelihood function of node representations via gradient ascent to update cluster centroids, where a differentiable diversity regularization term is introduced into the gradient function to enforce the diversity between cluster centroids, thereby increasing inter-cluster variance.
- **Better clustering performance.** We have empirically demonstrated that the proposed method can learn diverse and compact clusters, and achieve state-of-the-art clustering performance. The Ablation study confirms the effectiveness of each technique. Furthermore, we also conducted experiments to verify that our DDGC is not sensitive to the pretraining of the encoder and effectively alleviates the over smoothing issue.

The paper is structured as follows: Section 2 presents recent advancements in deep graph clustering; Section 3 explains the proposed DDGC for graph clustering in detail; Section 4 reports on the comprehensive experimental study. Finally, Section 5 provides the conclusion of the paper.

2. Related work

This section provides a review of the relevant literature. Deep clustering seeks to employ neural networks to learn deep feature representations favoring the clustering tasks. We focus on the node-level graph clustering task. With advancements in graph representation learning, significant improvements have been achieved in graph clustering. Existing algorithms for deep graph clustering can be categorized into two groups: (i) *Two-step Methods* and (ii) *Fine-tuning Methods*.

2.1. Two-step methods

Two-step Methods first train a graph encoder to learn node representations via minimizing an unsupervised loss. Subsequently, they perform conventional clustering methods, such as k-means and spectral learning, on the learned node representations to obtain clustering results. The key differences among them lie in the choice of graph representation learning models, which include the variants of graph Auto-Encoder and graph contrastive learning models.

Proper design of the graph convolution layer is a key factor in improving clustering performance. To employ GCN in graph autoencoder, GAE [28] was proposed first to transform each node into latent representations via GCN and then reconstruct the adjacency matrix via the decoder. [15] implemented a sparse autoencoder model that penalizes both the reconstruction error and the sparsity error in the hidden layer for learning non-linear representations for the clustering. [29] further improved the sparse autoencoder by enforcing a constraint of KL divergence between the distribution of node attributions on clusters and predicted distribution by the neural network for better clustering

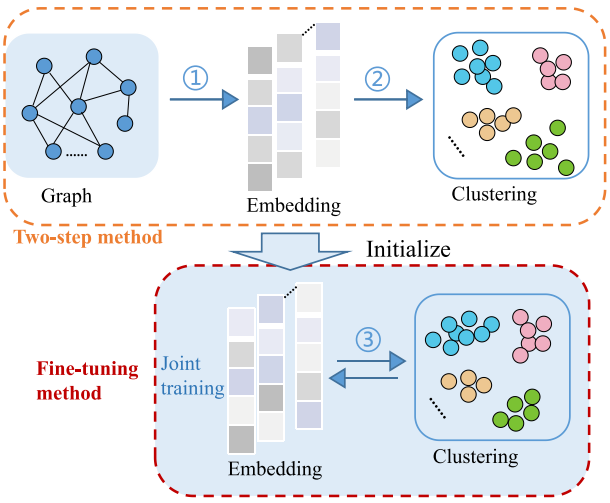


Fig. 4. The comparison between the *Two-step Methods* and *Fine-tuning Methods*. In the two-step method, the first step ① aims to learn node representations via graph autoencoder. Based on the learned representations, the second step ② aims to learn cluster centroids via performing the conventional clustering methods, *e.g.*, K-means or spectral method. Besides, the existing *Fine-tuning Methods* typically use the graph encoder learned by the two-step method as initialization. ③ aims to jointly train cluster centroids and node representations.

performance. [19] proposed an adversarial regularization to enforce the representations to match a prior distribution for learning compact representations and then applied k-means on the representations. [30] developed a denoising strategy to reduce redundant information and noise to enhance the robustness of the autoencoder. [31] proposed a marginalized graph convolutional network to corrupt network node content and marginalized the corrupted features to learn graph feature representations, upon which spectral clustering was applied for clustering.

Following the success of contrastive learning in graph and computer vision, researchers have recently investigated how to combine contrastive learning and traditional clustering methods. Specifically, [20] adopted contrastive learning to learn unsupervised node representations, instead of graph autoencoder, where the positive samples of the query node are designed to be randomly drawn from the cluster to which the query belongs, while the negative samples are randomly sampled from other clusters. However, the lack of adequate task-orientated representation learning always limits the performance of *Two-step Methods*. That is, unsupervised representation learning models are not designed dedicatedly for graph clustering. Thus, the clustering may be misguided by the representation learning, leading to non-optimal results.

2.2. Fine-tuning methods

To achieve the mutual benefit between these two steps, *Fine-tuning Methods* propose to minimize a designed KL-based clustering loss for simultaneously training node representations and cluster centers during the fine-tuning stage. The main challenge for crafting *Fine-tuning Methods* lies in devising a well-thought-out clustering loss and establishing effective self-supervised strategies for node clustering.

In particular, after pretraining a graph encoder and centroids, DAEGC [24] simultaneously optimized node representations and centroids by minimizing the designed task-oriented clustering loss in the fine-tuning stage. It adopted the second power of the predicted clustering assignment distribution of each node as the target distribution. The KL-divergence between these two distributions served as the clustering loss to jointly fine-tune the node representations and cluster centroids. The peaked target distribution acted as the self-generated

pseudo-label to supervise the learning of node clustering. Guided by "confident" assignments as soft labels, this KL-based clustering loss can drive nodes closer to the cluster centers, leading to more compact clusters. SDCN [25] goes further by incorporating an additional clustering loss that evaluates the discrepancy between the soft clustering assignment distribution produced by GCN and its second power form, to facilitate the fine-tuning process. Meanwhile, DFCN [26] proposed an interdependency learning-based structure and an attribute information fusion module to explicitly merge the representations learned by an autoencoder and a graph autoencoder for consensus representation learning. DCRN [27] encoded different augmented graphs into representations using a siamese network, subsequently reducing information correlation at both sample-level and cluster-level respectively. It is worth noting that all these methods employ KL-divergence as the clustering loss. All those methods typically treat cluster centroids as parameters and fine-tune them with graph encoder via minimizing the KL-divergence clustering loss, where the graph encoder and cluster centroids are firstly pretrained using an *Two-step Method*. We refer to this type of methods as *Fine-tuning Methods*.

However, their success hinges upon the selection of a high-quality pretrained graph encoder and properly initialized cluster centroids. Their clustering performance often exhibits high sensitivity to the pretrained model, as empirically depicted in Fig. 2. Only when the high quality of the graph encoder is selected as the initialization in the fine-tuning stage, the clustering performance can be roughly guaranteed. However, in practical tasks, determining a reliable stopping criterion for selecting the pretrained encoder can be challenging. Therefore, joint training of encoders and centroids from scratch is typically necessary.

Different from the above *Two-step* and *Fine-tuning Methods* for the node-level graph clustering task we focus on, [32] introduced a graph-level clustering method, Deep Graph-level Clustering (DGLC). DGLC employed a graph isomorphism network to learn graph-level representations by maximizing mutual information between the representations of entire graphs and substructures. [33] devised a general graph clustering method applicable to four different graph types, encompassing directed, undirected, heterogeneous, and hyperattributed graphs. [34] mainly focused on developing a dual variational autoencoders by maximizing a derived variational lower bound. In this paper, we mainly design a effective graph clustering framework to iteratively achieve these two principles of maximizing inter-cluster variances while minimizing intra-cluster variances, instead of diligently improving the graph autoencoder. Besides, numerous existing studies have also focused on the task of multi-view clustering task [35–38]. [39] developed a GCN-based multi-view clustering network with weighting mechanism and collaborative training (DMVGC) over the graph and image data with a specially designed attention module in the encoder and an adaptive weighting mechanism in the decoder for the reconstruction loss. [8] designed an anchor-based multi-view graph clustering framework, Efficient Multi-View Graph Clustering with Local and Global Structure Preservation (EMVGC-LG) to optimize anchor construction and graph learning to enhance the clustering quality. Additionally, the topic of overlapping graph clustering was studied in [40], where a new overlapping graph clustering algorithm was proposed that integrates the topological structure and attributes into the cluster potential game. In computer vision, [7] introduced a new contrastive clustering algorithm with graph consistency constraint to reduce the effect of the uncertainty of positives and negatives on contrastive clustering for image datasets.

3. Proposed model

In this section, we will specifically describe our graph clustering model, **D**iversity-promoting **D**eep **G**raph **C**lustering model (DDGC). First, we introduce the Seeking-Diverse-Bellwether (SDB) module, where a diversity regularization term is incorporated into the gradient ascent formulation of the maximum log-likelihood function to promote diversity among the cluster centroids, thereby obtaining diverse cluster centroids. Furthermore, we introduce a Sharpening Clustering Assignment (SCA) module which utilizes the minimum entropy (ME) of node assignment distribution as the clustering loss instead of the conventional KL-divergence. Our ME-based clustering loss minimizes the intra-cluster variance by enforcing a one-hot clustering assignment distribution. The SDB module serves as the inner loop to learn the diverse centroids based on the learned node representations. In the outer loop, with the centroids fixed, the SCA module trains node representations by minimizing our ME-based clustering loss and the reconstruction loss. The overall structure of our proposed DDGC is illustrated in Fig. 5.

Basic Notations. An undirected attributed graph is represented as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, where $\mathcal{V} = \{v_i\}_{i=1}^n$ denotes the set of nodes with $|\mathcal{V}| = n$, $\mathcal{E}$ represents the set of edges, and $\mathbf{X} \in \mathbb{R}^{n\times d}$ indicates the feature matrix of nodes where d is the feature dimension associated with each vertex. The topological structure of the graph is represented as the adjacent matrix $\mathbf{A} \in \{0,1\}^{n\times n}$, where $\mathbf{A}_{ij} = 1$ if there exists an edge between nodes v_i and v_j; $\mathbf{A}_{ij} = 0$ otherwise. The goal of graph clustering is to partition all nodes into the m groups, each with its respective cluster centroids $\{\boldsymbol{\mu}_i\}_{i=1}^m$, in the latent space. Node representations $\mathbf{Z} \in \mathbb{R}^{n\times d}$ can be obtained via training a graph encoder.

3.1. Seeking diverse bellwethers

In this subsection, we specifically introduce the Seeking-Diverse-Bellwethers (SDB) module, which aims to learn diverse cluster centroids from the node representations obtained by the graph encoder. Recall that the deep graph clustering task aims to allocate n nodes into m categories in the latent space. Suppose that node representations $\mathbf{z} \in \{\mathbf{z}_i\}_{i=1}^n$ are drawn from a *simplified* multivariate Gaussian Mixture Model with the probability density function: $p(\mathbf{z}|\boldsymbol{\Theta}) := \frac{1}{m}\sum_{i=1}^m \mathcal{N}(\mathbf{z}|\boldsymbol{\theta}_i)$, where $\boldsymbol{\theta}_i = \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}$ denotes the parameters of the ith Gaussian with mean vector $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$. Its connection with the Gaussian Mixture Model (GMM) has been discussed in Remark 1. Assuming that all nodes are independent and identically distributed, the model parameters $\boldsymbol{\Theta}$ can be estimated by maximizing the log-likelihood function of all node representations, *i.e.*, $\hat{\boldsymbol{\Theta}} = \text{argmax}_{\boldsymbol{\Theta}} \log(\prod_{i=1}^n p(\mathbf{z}_i|\boldsymbol{\Theta}))$, where $\prod_{i=1}^n p(\mathbf{z}_i|\boldsymbol{\Theta})$ is the joint probability distribution. The log-likelihood function can be derived as:

$$F(\mathbf{Z}|\boldsymbol{\Theta}) = \log\left(\prod_{i=1}^n p(\mathbf{z}_i|\boldsymbol{\Theta})\right) = \frac{1}{n}\sum_{j=1}^n \log\Big(\frac{1}{m}\sum_{i=1}^m \mathcal{N}(\mathbf{z}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\Big) \tag{2}$$

where $\mathcal{N}(\mathbf{z}_j|\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma}_i)}}\exp[-\frac{1}{2}(\mathbf{z}_j-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\mathbf{z}_j-\boldsymbol{\mu}_i)]$ represents the probability of $\mathbf{z}_i$ conforming to a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. It is worth noting that the learnable mean vectors $\{\boldsymbol{\mu}_i\}_{i=1}^m$ are the cluster centroids we aim to find.

Remark 1 (*The Connection with Multivariate Gaussian Mixture Model (GMM)*)**.** In a multivariate GMM, the probability density function of observations is defined as $f(\mathbf{y}|\boldsymbol{\Theta}) = \sum_{i=1}^m \pi_i \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ with $\sum_i^m \pi_i = 1$. Given a set of independent observations $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^n$, the goal of multivariate GMM is to learn the parameters of multivariate normal distribution $\boldsymbol{\Theta} = \{(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\}_{i=1}^m$ and the weights $\{\pi_i\}_{i=1}^m$, via maximizing the following log-likelihood function,

$$F_{gmm}(\mathbf{Y}|\boldsymbol{\Theta}) = \frac{1}{n}\sum_{j=1}^n \log\Big(\sum_{i=1}^m \pi_i \mathcal{N}(\mathbf{y}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\Big). \tag{3}$$

Notably, optimizing both $\{\pi_i\}_{i=1}^m$ and $\boldsymbol{\Theta}$ directly is intractable, so the EM algorithm [41] is resorted to updates $\{\pi_i\}_{i=1}^m$ and $\boldsymbol{\Theta}$ by alternating the E-step (Expectation-step) and the M-step (Maximization-step), respectively. When $\pi_i = \frac{1}{m}, \forall i \in \{1, \ldots, m\}$, Eq. (3) is equivalent Eq. (2), namely, our clustering objective is a special case of GMM. With the assumption of $\pi_i = \frac{1}{m}$, each cluster nearly contains an equal number of nodes. This assumption of balanced clustering does not adversely impact our clustering performance. In this case, the model parameters

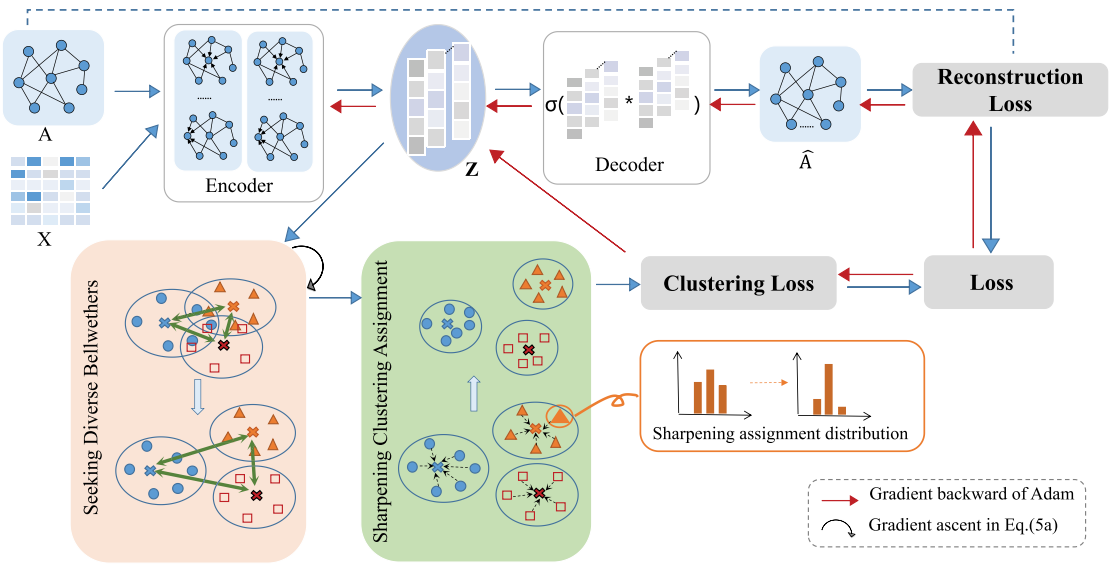


Fig. 5. Overview of the proposed DDGC framework. The graph encoder takes the adjacency matrix $\mathbf{A}$ and feature matrix $\mathbf{X}$ as the inputs and outputs node representations $\mathbf{Z}$. Based on the learned representations, the Seeking-Diverse-Bellwethers (SDB) module learns diverse cluster centroids by the gradient ascent in Eq. (4a). Then, the proposed Sharping Clustering Assignment (SCA) module computes the clustering assignment distribution of each node and leverages these distributions to calculate the ME-based clustering loss. The total loss, weighting the clustering loss and the reconstruction loss, is minimized to update the graph encoder via stochastic gradient descent while keeping the centroids fixed. The flow of gradient ascent is denoted as the red arrows. The grey arrows indicate the gradient ascent in Eq. (4a).

$\boldsymbol{\Theta}$ can be directly optimized via gradient ascent to maximize the log-likelihood function in Eq. (2) instead of the iterative optimization of the complex EM algorithm.

In general, the model parameters $\boldsymbol{\Theta}$ can be estimated via maximizing the log-likelihood function $F(\mathbf{Z}|\boldsymbol{\Theta})$ in Eq. (2). To promote the diversity among centroids, we perform the gradient ascent to update the cluster centroid $\{\boldsymbol{\mu}_i\}_{i=1}^m$, where a diversity regularizer is incorporated into the gradient function inspired by nonlinear stein variation gradient descent [42]. More precisely, we perform the following gradient ascent to optimize the cluster centroids until convergence:

$$\boldsymbol{\mu}_i^{t+1} = \boldsymbol{\mu}_i^t + \varepsilon\phi(\boldsymbol{\mu}_i^t), \quad \forall i = 1, \dots, m, \tag{4a}$$

$$\phi(\boldsymbol{\mu}_i^t) = \sum_{j=1}^{m} \left[\nabla_{\boldsymbol{\mu}_j^t} F(\mathbf{Z}|\boldsymbol{\Theta}) \cdot k(\boldsymbol{\mu}_j^t, \boldsymbol{\mu}_i^t) + \frac{\alpha}{m} \nabla_{\boldsymbol{\mu}_j^t} k(\boldsymbol{\mu}_j^t, \boldsymbol{\mu}_i^t)\right], \tag{4b}$$

where $\phi(\boldsymbol{\mu}_i^t)$ is the gradient of the centroid of cluster i at the tth iteration, ε is the small step size, the function $k(\cdot,\cdot)$ can be any positive definite kernel measuring the similarity between two centroids, and α is the balance coefficient to control the regularization strength. In Eq. (4a), for each cluster $i \in \{1, \dots, m\}$, we can perform the gradient ascent to obtain the centroid $\boldsymbol{\mu}_i^{t+1}$ of cluster i at the $(t+1)$-th step. Eq. (4b) is the expansion equations of the gradient $\phi(\boldsymbol{\mu}_i^t)$. More specifically, in Eq. (4b), the first term aims to fit the given node representations by increasing the log-likelihood function. The second term is a diversity regularizer, which acts as a repulsive force to prevent all cluster centroids from collapsing together into local modes. More detailed elaboration on the roles of these two terms in Eq. (4b) is provided below:

- **Fitting representations.** The first term in Eq. (4b) is instrumental in maximizing the log-likelihood function $F(\mathbf{Z}|\boldsymbol{\Theta})$ through gradient ascent in Eq. (4a), whose role is to fit the given node representations $\mathbf{Z} \in \mathbb{R}^{n\times d}$. More specifically, the first term computes weighted gradients of the log-likelihood function $F(\mathbf{Z}|\boldsymbol{\Theta})$ to each cluster centroid $\{\boldsymbol{\mu}_j^t\}_{j=1}^m$. These weight are determined by the similarity $k(\boldsymbol{\mu}_i^t, \boldsymbol{\mu}_j^t)$ between the centroids of cluster i and the cluster j, $j \in \{1, \dots, m\}$. The gradient of $F(\mathbf{Z}|\boldsymbol{\Theta})$ to each cluster centroid $\boldsymbol{\mu}_j^t$ can be derived as:

$$\nabla_{\boldsymbol{\mu}_j^t} F(\mathbf{Z}|\boldsymbol{\Theta}) = \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{\Sigma}_j^t)^{-1}(\mathbf{z}_i - \boldsymbol{\mu}_j^t), \tag{5}$$

  It is noteworthy that the gradients of all centroids, $\nabla_{\boldsymbol{\mu}_j^t} F(\mathbf{Z}|\boldsymbol{\Theta}), j \in \{1, \dots, m\}$ are used to update $\boldsymbol{\mu}_i^{t+1}$, but the gradient of $\boldsymbol{\mu}_i^t$ remains dominant its own updates at the $(t+1)$-th iteration, because $k(\boldsymbol{\mu}_i^t, \boldsymbol{\mu}_i^t) = 1 \geq k(\boldsymbol{\mu}_i^t, \boldsymbol{\mu}_j^t), \forall j \neq i$. Moreover, if $\boldsymbol{\mu}_j^t$ is closer to $\boldsymbol{\mu}_i^t$ than other centroids, $\nabla_{\boldsymbol{\mu}_j^t} F(\boldsymbol{\Theta})$ contributes more significantly to the update of $\boldsymbol{\mu}_i^t$, due to the sharing of more semantic information.

- **Diversifying centroids.** The second term in Eq. (4b) serves as a repulsive force to promote diversity among the centroids $\{\boldsymbol{\mu}_i\}_{i=1}^m$. We adopt the Radial Basis Function (RBF) kernel as a specific example of $\phi(\boldsymbol{\mu}_i^t)$ while noting that many other choices are applicable. The RBF kernel is expressed as:

$$k(\boldsymbol{\mu}, \boldsymbol{\mu}') = \exp(-\frac{\|\boldsymbol{\mu} - \boldsymbol{\mu}'\|^2}{2h^2}),$$

  where h denotes the bandwidth, typically set to the median of the distance between pairwise centroids. Therefore, the gradient of $k(\boldsymbol{\mu}_j^t, \boldsymbol{\mu}_i^t)$ to $\boldsymbol{\mu}_j^t$ in Eq. (4b) can be derived as:

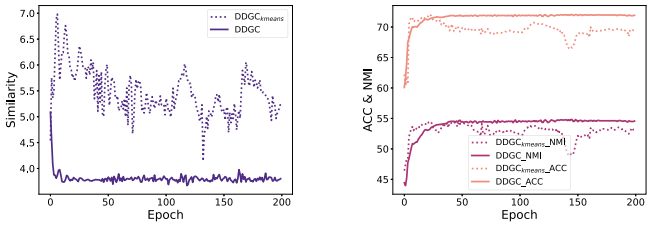
$$\nabla_{\boldsymbol{\mu}_j^t} k(\boldsymbol{\mu}_j^t, \boldsymbol{\mu}_i^t) = -\frac{|\boldsymbol{\mu}_j^t - \boldsymbol{\mu}_i^t|}{h^2} \exp(-\frac{\|\boldsymbol{\mu}_j^t - \boldsymbol{\mu}_i^t\|^2}{2h^2}), \tag{6}$$

  which plays a crucial role in promoting diverse centroid learning. We refer to it as the diversity regularizer. Specifically, if the neighboring centroid $\boldsymbol{\mu}_j^t$ is closer to $\boldsymbol{\mu}_i^t$ at the t-the iteration, the gradient $\nabla_{\boldsymbol{\mu}_j^t} k(\boldsymbol{\mu}_j^t, \boldsymbol{\mu}_i^t)$ provided by $\boldsymbol{\mu}_j^t$ will be larger. Consequently, this larger gradient will push $\boldsymbol{\mu}_i^t$ and $\boldsymbol{\mu}_j^t$ away from each other at the $(t+1)$-th iteration. In essence, the closer neighboring centroid significantly influences the update of $\boldsymbol{\mu}_i^t$. Ultimately, this process results in the diversification and uniform separation of all cluster centers within the latent space.

Metric of diversity. The diversity regularizer is incorporated into the gradient function in Eq (4b) for enhancing the diversity among cluster centroids. To provide a more intuitive insight into the degree of separation among centroids, we propose the following metric to quantify the diversity of centroids:

$$D = \mathbb{E}_{i\neq j} k(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) = \mathbb{E}_{i\neq j} \exp(-\frac{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2}{2h^2}),$$

where D represents the expectation of the kernel values across all pairs of centroids. A lower value of D indicates larger diversity and less



(a) Similarity vs. Epoch (b) ACC (%) & NMI (%)

Fig. 6. DDGC *vs.* DDGC$_{kmeans}$ on Cora. (a) DDGC exhibits lower similarity and greater diversity between cluster centroids compared to DDGC$_{kmeans}$. (b) DDGC achieves better clustering performance in terms of ACC and NMI.

similarity among centroids. To empirically investigate the impact of diversifying centroids, we conduct comparative experiments between our model, DDGC, and its variant, DDGC$_{kmeans}$, on the Cora dataset, which replaces the seeking-diverse-bellwethers (SDB) module with kmeans in DDGC. The clustering results are illustrated in Fig. 6. With the increase of epochs, the similarity/diversity among centroids in both DDGC and DDGC$_{kmeans}$ decrease/increase gradually. However, the SDB module enables DDGC to achieve lower similarity scores, alongside higher ACC (Accuracy) and NMI (Normalized Mutual Information), which demonstrates our DDGC can learn more diversified centroids compared to DDGC$_{kmeans}$, contributing to the improved clustering performance.

3.2. Sharpening clustering assignment

In this subsection, we introduce the Sharpening Clustering Assignment (SCA) module to minimize intra-cluster variances. The Minimum Entropy (ME)-based clustering loss is proposed to replace the conventional KL-based clustering loss, and its advantage will be explained by comparing these two losses. Furthermore, we compare the gradients of those two clustering losses to the clustering assignment distribution, which verifies theoretically the ME-based loss exhibits larger gradients, enabling the assignment distribution to be sharper.

3.2.1. Proposed ME-based clustering loss

The design of the clustering loss has always been a challenge for the unsupervised clustering task, because of the difficulty of designing self-supervised labels. In this paper, we propose a new method that get rid of the necessity of designing self-supervised labels. Our method employs minimum entropy as the clustering loss to sharpen the clustering assignment distribution. Specifically, to obtain the clustering assignment distribution of each node, we utilize Student's t-distribution to estimate the probability of node i belonging to cluster j, via quantifying the normalized similarity between the node representation $\mathbf{z}_i$ and the cluster centroid $\boldsymbol{\mu}_j$:

$$p_{ij} = \frac{(1 + \|\mathbf{z}_i - \boldsymbol{\mu}_j\|^2/a)^{-\frac{a+1}{2}}}{\sum_{j=1}^{m}(1 + \|\mathbf{z}_i - \boldsymbol{\mu}_j\|^2/a)^{-\frac{a+1}{2}}}, \tag{7}$$

where m is the number of clusters, a denotes the degree of freedom of the Student's t-distribution, which is set to 1 in experiments. Besides, $p_{ij} \in [0, 1]$ indicates the probability of assigning node i to cluster j. For any node i, its assignment distribution can be represented by a m-dimensional vector $\mathbf{p}_i = [p_{i1}, \ldots, p_{im}]^{1\times m}$ with $\sum_j p_{ij} = 1$. The assignment distributions of all n node can form the predicted assignment matrix, $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_n]^{n\times m}$. Furthermore, our proposed ME-based clustering loss over the node assignment distribution can be formulated as follows:

$$L_e = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}\log p_{ij}. \tag{8}$$

Minimizing the clustering loss L_e can sharpen the node assignment distributions toward the one-hot distribution, thereby promoting the compactness of clusters and clearer boundaries, ultimately enhancing clustering performance. To illustrate the advantage of our Minimum Entropy (ME)-based clustering loss, we will compare it with the conventional KL-based clustering loss, which will be introduced in detail next.

Conventional KL-based clustering loss. The KL-based clustering loss commonly used in previous literature [24,25] is defined as the KL-divergence distance between the predicted assignment distribution $\mathbf{P} \in [0, 1]^{n\times m}$ and the self-generated target distribution $\mathbf{Q} \in [0, 1]^{n\times m}$ across all nodes, expressed as:

$$L_{kl} = KL(\mathbf{P} \parallel \mathbf{Q}) = \frac{1}{n}\sum_{i}^{n}\sum_{j}^{m} q_{ij}\log\frac{q_{ij}}{p_{ij}}, \tag{9}$$

where $q_{ij} = \dfrac{p_{ij}^2/\sum_i p_{ij}}{\sum_u (p_{iu}^2/\sum_i p_{iu})}$.

Here, the target probability q_{ij} is calculated by squaring and normalizing the predicted assignment probability p_{ij}, which serves as the pseudo-label to supervise the training of the graph encoder. Thus, the target distribution of node i, $\mathbf{q}_i = [q_{i1}, \ldots, q_{im}]^{1\times m}$, is a peaker distribution, which forms the target assignment matrix of all nodes, $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_n]^{n\times m}$. Minimizing the KL-based clustering loss in Eq. (9) enables the predicted assignment distribution $\mathbf{p}_i$ approaching to the target distribution $\mathbf{q}_i$ for any node i, which can effectively pull node i to its centroids, and thus promote cluster compactness. Next, we derive the gradient of L_{kl} to the cluster centroid $\boldsymbol{\mu}_j$ as follows:

$$\nabla_{\boldsymbol{\mu}_j} L_{kl} = -\frac{a+1}{a}\sum_i (1 + \frac{\|\mathbf{z}_i - \boldsymbol{\mu}_j\|^2}{a})^{-1} \times (p_{ij} - q_{ij})(\mathbf{z}_i - \boldsymbol{\mu}_j).$$

Since the target probability, q_{ij} acts as the pseudo-label, it is regarded as a constant at each epoch. The gradient $\nabla_{\boldsymbol{\mu}_j} L_{kl}$ includes an extra term $(p_{ij} - q_{ij})$, which implies that the optimization direction of the cluster center $\boldsymbol{\mu}_i$ may be influenced by the pre-defined q_{ij}, may potentially leading to the learning of incorrect cluster centroids. Specifically, if the predicted assignment distribution p_{ij} is inaccurate, the manipulation of squaring p_{ij} can magnify the error, resulting in an suboptimal, even incorrect optimizing direction, such that the pseudo-label q_{ij} may misguide the training of centroids and graph model. But our ME-based loss does not rely on any self-generated target distribution, so it is more flexible to correct the error of p_{ij}, effectively addressing the limitation of the convention KL-based loss.

Remark 2 (*The Conventional KL-based Clustering Loss Relies on the Pre-Defined Pseudo-Label, Potentially Leading to Suboptimal Centroid Optimization*)**.** Obviously, in Eq. (8), our ME-based clustering loss L_e depends solely on the predicted assignment probability p_{ij}, thereby getting rid of requirement of designing pre-defined supervision. Next, we derive the gradient of the KL-based loss L_{kl} to the cluster centroid $\boldsymbol{\mu}_j$ as outlined below:

$$\nabla_{\boldsymbol{\mu}_j} L_{kl} = -\frac{a+1}{a}\sum_i (1 + \frac{\|\mathbf{z}_i - \boldsymbol{\mu}_j\|^2}{a})^{-1} \times (p_{ij} - q_{ij})(\mathbf{z}_i - \boldsymbol{\mu}_j).$$

where the target probability, q_{ij}, acting as the pseudo-label, is regarded as a constant at each epoch. The gradient $\nabla_{\boldsymbol{\mu}_j} L_{kl}$ can be updated by using feature-space embeddings $\{\mathbf{z}_i\}_{i=1}^{n}$. However, it also includes an extra term $(p_{ij} - q_{ij})$, indicating that the optimization direction of the cluster center $\boldsymbol{\mu}_i$ can be influenced by the pre-defined p_{ij}, may potentially leading to the learning of incorrect cluster centroids.

3.2.2. A gradient comparison between the ME- and KL-based clustering loss

To further elucidate the superiority of our ME-based clustering loss, we conduct a theoretical comparison of the gradients of the ME- and KL-based clustering losses to the assignment probability. These gradients propagated through the chain rule, play a crucial role in updating the graph encoder. For simplicity, we take a clustering task

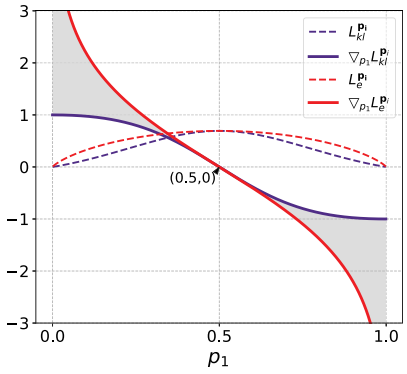


Fig. 7. The comparison of the gradients in ME- and KL-based clustering loss. Under any gradient, if $p_1 < 0.5$, p_1 will be updated toward 0; if $p_1 > 0.5$, p_1 will be updated toward 1. The gradient of ME $\nabla_{p_1} L_e^{\mathbf{p}_i}$ (solid red line) consistently exceeds that of KL $\nabla_{p_1} L_{kl}^{\mathbf{p}_i}$ (solid blue line). And $p_1 \to \{0, 1\}$, the gap (grey area) between two gradients widens, leading to a faster update of p_1 toward 0/1 with the gradient of ME-based loss .

with two clusters as an example. Let $\mathbf{p}_i = [p_1, 1 - p_1]$ denote the assignment distribution of node i. The ME-based clustering loss over $\mathbf{p}_i$ can be derived as:

$$L_e^{\mathbf{p}_i} = -[p_1 \log p_1 + (1 - p_1) \log(1 - p_1)], \tag{10}$$

where p_1 and $(1 - p_1)$ denotes the probabilities of assigning node i to the cluster one and cluster two, respectively. Then, the gradient of $L_e^{\mathbf{p}_i}$ to p_1 can be calculated as follows:

$$\nabla_{p_1} L_e^{\mathbf{p}_i} = -\log \frac{p_1}{1 - p_1}, \tag{11}$$

which is the function of only one variable p_1. Next, we calculate the gradient of the KL-based clustering loss to p_1. For a clustering task with two clusters, the KL-divergence clustering loss in Eq. (9), can be rewritten as,

$$L_{kl}^{\mathbf{p}_i} = \underbrace{q_1 \log q_1 + (1 - q_1) \log(1 - q_1)}_{Constant} \underbrace{-q_1 \log p_1 - (1 - q_1) \log(1 - p_1)}_{Reduction\ term}. \tag{12}$$

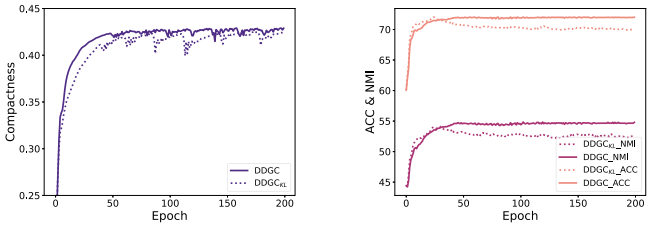
where q_1 is the target probability of node i belonging to cluster one, $q_1 = \frac{p_1^2}{p_1^2 + (1-p_1)^2}$[1]. Due to serving as the pseudo-label of p_1, q_1 remains fixed and does not participate in the gradient back-propagation. Thus $L_{kl}^{\mathbf{p}_i}$ can be split into the sum of two terms, the constant and reduction term. Only the reduction term contains the variable p_1, so it can be used to optimize model parameters. The gradient of $L_{kl}^{\mathbf{p}_i}$ to p_1 can be derived as:

$$\nabla_{p_1} L_{kl}^{\mathbf{p}_i} = -\frac{q_1}{p_1} + \frac{1 - q_1}{1 - p_1} = \frac{1 - 2p_1}{p_1^2 + (1 - p_1)^2}. \tag{13}$$

These two gradients, $L_{kl}^{\mathbf{p}_i}$ and $L_e^{\mathbf{p}_i}$ to p_1, are both the functions of the variable p_1. Hence, we depict the two gradients[2] in Eqs. (11) and (13), as well as the functions of $L_e^{\mathbf{p}_i}$ and the reduction term of $L_{kl}^{\mathbf{p}_i}$ in Fig. 7. As observed, $\nabla_{p_1} L_{kl}^{\mathbf{p}_i}$ has an upper bound: $\nabla_{p_1} L_{kl}^{\mathbf{p}_i} \leq 1$ with $\lim_{p_1 \to 0/1} \nabla_{p_1} L_{kl}^{\mathbf{p}_i} = 1$. In contrast, $\nabla_{p_1} L_e^{\mathbf{p}_i}$ has no bound, and $\lim_{p_1 \to 0/1} \nabla_{p_1} L_e^{\mathbf{p}_i} = \infty$. With p_1 approaching 0/1, the gradient of the ME-based clustering loss significantly surpasses that of the KL-based clustering loss. Thus, $\nabla_{p_1} L_e^{\mathbf{p}_i}$ can push the assignment distribution of node i, $[p_1, 1 - p_1]$, closer to the one-hot distribution compared to $\nabla_{p_1} L_{kl}^{\mathbf{p}_i}$, thereby verifying the advantage of sharpening the assignment distribution of our ME-based clustering loss. Therefore, our ME-based clustering loss can produce more compact clusters than the KL-based loss. To quantify the compactness of the cluster, we propose the following metric to observe the differences between the clusters produced by those two losses.

[1] In Eq. (9), both $\sum_i p_{ij}$ and $\sum_i p_{iu}$ in the numerator and denominator are the normalization terms, representing the sum of assignment probability within clusters, which is used to enforce balanced clustering assignment. As we focus on balance clustering, these two terms can be omitted in derivations.

[2] In Fig. 7, the "log" in L_e, L_{kl}, $\nabla_{p_1} L_e^{\mathbf{p}_i}$, and $\nabla_{p_1} L_{kl}^{\mathbf{p}_i}$ are based on e, *i.e.*, ln, which is consistent with our source code.



(a) Compactness vs. Epoch (b) ACC (%) and NMI (%)

Fig. 8. DDGC *vs.* DDGC$_{KL}$ on Cora. (a) DDGC produces more compact clusters than DDGC$_{KL}$ because of the larger compactness metric across all epochs. (b) DDGC consistently outperforms DDGC$_{KL}$ in terms of both ACC and NMI.

Metric of Compactness. For the clustering task of m clusters, we utilize the average maximum probability of each assignment distribution $\mathbf{p}_i$ to measure the compactness of the clusters:

$$C = \frac{1}{n} \sum_{i=1}^{n} p_i^*, \quad \text{where } p_i^* = \max \mathbf{p}_i$$

where p_i^* represents the maximum probability of assigning node i to clusters, indicating the confidence of the node belonging to its own centroid. C denotes the averaged assignment confidence over all nodes. A higher C signifies a lower distance from nodes to their own centroids and a higher compactness of clusters. To investigate the effectiveness of our ME-based loss, we conduct experiments to compare our model DDGC and its variant DDGC$_{KL}$, which replaces ME with KL-divergence. As depicted in Fig. 8, with the increase of the training epochs, the compactness value of both ME- and KL-based clustering losses gradually increases. However, benefiting from the larger gradient, our ME-based loss with larger C can produce more compact clusters and achieve better clustering performance compared to the KL-based loss.

3.3. The whole framework

In this section, we demonstrate the framework of the proposed DDGC, which incorporates these two modules: Seeking Diverse Bellwethers (SDB) and Shapening Clustering Assignment (SCA), to jointly optimize node representations and clustering. The overall framework is illustrated in Fig. 5. The graph encoder takes the adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ and the feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d_0}$ as inputs to learn node representations $\{\mathbf{z}_i\}_{i=1}^m$. Based on the node representations, the SDB module maximizes the log-likelihood function to learn the diverse cluster centroids, $\{\boldsymbol{\mu}_i\}_{i=1}^m$, through gradient ascent, as described in Eq. (4a). Based on the centroids, the SCA module calculates the clustering assignment distribution of each node and our ME-based clustering loss, which is minimized together with the reconstruction loss to update the graph encoder via stochastic gradient descent.

Graph AutoEncoder. Following DAEGC [24], we adopt the graph attentional autoencoder to learn the node representations by minimizing the reconstruction loss. The encoder takes both the graph structure and node attributes as inputs, which outputs the representation $\mathbf{z}_i^{l+1}$ at the l layer by aggregating the neighbors $\mathcal{N}_i$ of node i:

$$\mathbf{z}_i^{l+1} = \sigma\Big(\sum_{j \in \mathcal{N}_i} \beta_{ij} \mathbf{W}^l \mathbf{z}_j^l\Big), \tag{14}$$

where $\beta_{ij} = \frac{\exp(\delta M_{ij}(\vec{h}^T[\mathbf{W}^l\mathbf{z}_i^l \parallel \mathbf{W}^l\mathbf{z}_j^l]))}{\sum_{r\in\mathbb{N}_i}\exp(\delta M_{ir}(\vec{h}^T[\mathbf{W}^l\mathbf{z}_i^l \parallel \mathbf{W}^l\mathbf{z}_r^l]))}$

where σ is a nonlinear function, $\mathbf{W}^l$ denotes the network parameters of the lth layer, and β_{ij} is the attention coefficient indicating the importance of neighbor j to node i. Additionally, δ is an activation function, and $\vec{h} \in \mathbb{R}^{2d}$ is a parameter vector. $M = (B + B^2 + \cdots + B^t)/t$ denotes a proximity matrix obtained by exploiting t-order neighbor nodes. Here B is the transition matrix where $B_{ij} = \frac{1}{d_i}$ if $e_{ij} \in \mathcal{E}$ and $B_{ij} = 0$ otherwise. The decoder we adopt is a simple inner product $\hat{\mathbf{A}}_{ij} = \sigma(\mathbf{z}_i^\top \mathbf{z}_j)$, which predicts the probability of an edge between node i and node j and thus outputs the reconstructed adjacency matrix $\hat{\mathbf{A}}$. In principle, we could use any form of graph encoder and decoder to compute the graph reconstruction loss.

Total Loss. The graph autoencoder can be optimized via minimizing the total loss, which comprises the proposed clustering loss and the reconstruction loss. The reconstruction loss of the graph autoencoder is defined as the difference between the reconstructed adjacency matrix $\hat{\mathbf{A}}$ and the given adjacency matrix $\mathbf{A}$:

$$L_r = \sum_{i=1}^{n}\sum_{j=1}^{n} loss(\mathbf{A}_{i,j}, \hat{\mathbf{A}}_{i,j}). \tag{15}$$

The total loss can be computed by weighting the reconstruction loss L_r and the proposed ME-based clustering loss L_e:

$$L = L_r + \eta L_e, \tag{16}$$

where η is a balance factor. To control the increasing impact of ME-based loss on the total loss, we design a dynamic coefficient using a sigmoid-like function,

$$\eta(t; a, b, c) = \frac{c}{1 + e^{-at+b}}, \tag{17}$$

where t represents the epoch number. a, b, and c are hyperparameters to control the rate, initial weight, and amplitude, respectively. The curves of η with different a and b values are illustrated in Fig. 13 in Appendix. Overall, with the increase of t, η gradually grows regardless of the values of a and b. Our method DDGC is summarized in Algorithm 1.

Our DDGC method mainly consists of two components: clustering and representation optimization.

- **Clustering Optimization.** Given the node representations learned by the graph encoder, the seeking diverse bellwether (SDB) module performs the gradient ascent to maximize the log-likelihood function in Eq. (4a) for obtaining the diverse cluster centroids in the inner loop.
- **Representation Optimization.** Based on the learned centroids, the Sharpening Clustering Assignment module (SCA) calculates the clustering assignment distributions of all nodes and the ME-based clustering loss L_e. The total loss L, which weights the reconstruction loss and the ME-based clustering loss, is minimized to update the graph encoder and optimize node representations in the outer loop.

While minimizing the total loss L to update the graph encoder and optimize node representations, we keep the cluster centroids fixed in the outer loop. Conversely, we keep the representations fixed while maximizing the log-likelihood function $F(\mathbf{Z}|\boldsymbol{\Theta})$ for T epochs in the inner loop to optimize the cluster centroids via the gradient ascent in Eq. (4a). Essentially, our framework decomposes the optimization of graph encoder and cluster centroids, facilitating a more efficient and stable training procedure. However, the conventional KL-based clustering loss is often minimized to optimize the node representations and cluster centroids simultaneously, where centroids are taken as model parameters to be updated with the graph encoder in the fine-tuning stage. This coupling of optimization into the same stage prevents these methods from finding the optimal solution. The Algorithm 1 outlines the complete process of our DDGC. We first minimize the reconstruction loss for K' epochs for the pretrain graph encoder to encourage the encoder to explore better representations fully. Moreover, we also derive another version of DDGC without the pretraining stage for K' epochs, enabling DDGC to train from scratch, called DDGC *w/o pre*.

Algorithm 1 Algorithm of our DDGC.

Input: Graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$ with the set of nodes $\mathcal{V}$, the set of edges $\mathcal{E}$ and feature matrix $\mathbf{X}$; m: number of clusters; T: number of epochs for learning diverse centroids; K': number of epochs for pretraining representations; K: number of epochs for minimizing the total loss; l: number of layers of graph encoder; $\{\mathbf{W}^j\}_{j=1}^l$: the network parameters of l layers; α: the diversity hyperparameter (Eq. (4b)); a, b, c: hyperparameters in adaptively balance coefficient (Eq. (17)).
Output: The clustering results
for $0 \leftarrow k$ to $K' - 1$ **do**
  Update graph encoder $\{\mathbf{W}^j\}_{j=1}^l$ by minimizing the reconstruction loss in Eq. (15).
end for
for $K' \leftarrow k$ to K **do**
  Learn node representations $\{\mathbf{z}_i\}_{i=1}^n$ via the graph encoder.
  for $t = 1, 2, \ldots, T$ **do**
    Learn the diverse cluster centroids $\{\boldsymbol{\mu}_i\}_{i=1}^m$ by the gradient ascent Eq. (4a) and Eq. (4b)
  end for
  Calculate the clustering assignment distribution of each node based on the learned $\{\mathbf{z}_i\}_{i=1}^n$ and $\{\boldsymbol{\mu}_i\}_{i=1}^m$ via Eq. (7).
  Calculate the reconstruction loss L_r in Eq. (15) and the minimum entropy loss L_e in Eq. (8), respectively.
  Update graph encoder $\{\mathbf{W}^j\}_{j=1}^l$ by minimizing the total loss $L = L_r + \eta L_e$.
end for
Predict the cluster according to the maximum probability in the node clustering assignment distribution $s_i = arg\max_j p_{ij}$.
Evaluate the predicted clusters in terms of clustering metrics.
Return the clustering results.

3.4. Complexity analysis

In this subsection, we compare the proposed DDGC and state-of-the-art *Fine-tuning Methods* in terms of model and time complexity. In general, DDGC has fewer model parameters and lower computational complexity than baselines. To simplify, we assume the dimension of the hidden embeddings is d in all layers of the graph encoder, then the centroid dimension is d. The number of clusters is m. The number of edges is $|\mathcal{E}|$.

Model Complexity. Our DDGC has lower model parameters than the simplest fine-tuning method, DAEGC [24]. Specifically, DDGC adopts the same attentional Graph Autoencoder (GAE) as DAEGC to learn node representations. But in the fine-tuning stage, DAEGC takes the cluster centroids as the model parameters to be updated together with the graph encoder, which brings an additional $O(md)$ of model complexity than our DDGC. Moreover, there are two more complex *Fine-tuning Methods*, DFCN [26] and DCRN [27]. The graph encoder adopted by them includes three modules: GAE, Improved GAE (IGAE), and a fusion module, which will obviously bring larger model complexity. This complex encoder also needs to be fine-tuned, which raises higher model complexity in the fine-tuning stage.

Time Complexity. For our DDGC, in the inner loop, the computational complexity of the SDB module is $O(Tnmd^2)$, where its dominant term, nmd^2, arises from the computations of the multivariant Gaussian Mixture Model as described in Eq.(2) (its covariance matrix is an identity matrix with a diagonal of 1), and T denotes the number of training epoch of inner loop. Therefore, the total computational complexity of our DDGC is $O(K|\mathcal{E}|dk + (K-K')Tnmd^3)$, where K denotes the number of the total epoch, the sum of the pretraining epoch K' and fine-tuning epoch $(K - K')$. Another dominant term, $|\mathcal{E}|dk$, arises from

Table 1
The statistics of datasets.

Datasets	#Node	#Feature	#Clusters	#Links	Density
Cora	2708	1433	7	5429	0.074%
Citeseer	3327	3703	6	4732	0.043%
DBLP	4058	334	4	7056	0.043%
ACM	3025	1870	3	26,256	0.287%
AMAP	7650	745	8	238,163	0.407%

the attention-based graph encoder with k attention head, it training is across the total training process.

For DAEGC, the computation cost of its graph encoder is the same as that of our DDGC, *i.e.*, $|\mathcal{E}|dk$, since our DDGC follows the encoder of DAEGC. The computation cost of the Student-t distribution, $O(nmd)$, dominates the complexity of the clustering part of DAEGC. DAEGC pretrains the encoder for K' epochs and fine-tunes the cluster centroids and encoder for $(K-K')$ epochs, leading to time complexity, $O(K|\mathcal{E}|dk+(K-K')nmd)$. For DFCN and DCRN, both use two components as encoder, *i.e.*, Auto-Encoder and Improved Graph Auto-Encoder, and use deconvolution layers to reconstruct the node attributes and adjacency matrix. Compared with the simple Graph Auto-Encoder and inner-product decoder in DDGC and DAEGC, they have clearly introduced more computational complexity. For the clustering part in fine-tuning stage, the dominant term $O(n^2d)$ of DFCN comes from the normalized self-correlation matrix (Eq.(8) in [26]). The dominant term $O(n^3)$ of DCRN comes from two losses (Eqs. (5) and (8) in [27]) that makes the cross-view feature correlation matrix equal to an identity matrix. Therefore, the computational complexity of DFCN and DCRN are respectively $O(2K|\mathcal{E}|d+(K-K')n^2d)$ and $O(2K|\mathcal{E}|d+(K-K')n^3)$

To sum up, we can find that: i) Compared to DAEGC, our DDGC has achieved comparative time complexity due the only induced linear complexity of T and d^2 with $d \ll n$ and $T \ll n$; ii) Obviously, DFCN and DCRN exhibit higher time complexity than our DDGC and DAEGC; iii) Scaling to large-scale datasets only induced linear time complexity compared to DAEGC. TBesides, to ensure the convergence of the log-likelihood objective in the SDB module, we set the training epoch T to 300. However, T can be further reduced in practical applications.

4. Experiments

In this section, we evaluate the clustering performance of the proposed DDGC on five widely-used benchmark datasets. We compare our DDGC against 11 state-of-the-art (SOTA) methods to verify its superiority. Besides, we conduct an ablation study to demonstrate the effectiveness of each component and analyze the hyperparameter sensitivity. Furthermore, we validate that DDGC can alleviate the *over-smoothing* issue of the graph convolution network.

4.1. Benchmark datasets and baselines

To evaluate the effectiveness of the proposed DDGC model, we conducted extensive experiments on five widely-used benchmark graph datasets: Cora [24], Citeseer [28], DBLP[3], ACM[4], and AMAP [43]. Table 1 summarizes the structural details of the five graph datasets, where the density is defined as the number of existing edges divided by the square of the number of nodes, indicating how sparse a graph is.

For a comprehensive evaluation, we compared DDGC with the state-of-the-art methods from three categories:

- ***Two-step methods*** design various graph autoencoders to learn unsupervised node representations. Subsequently, based on the learned representations, they employ the traditional clustering method, such as kmeans and spectral clustering to obtain the clustering results. The differences among these methods primarily lie in the design of the graph autoencoder. These *Two-step Methods* mainly include GAE [28], VGAE [28], TADW [44], and ARGE [19], which design GAE, VGAE, DeepWalk, adversarially regularized (variational) graph autoencoder, respectively.
- ***Fine-tuning Methods*** including DAEGC [24], AGCN [45], SDCN [25], DFCN [26], and DCRN [27], AGCC [46], jointly train the cluster centroids and graph encoder by minimizing the KL-based clustering loss and the representation loss in the fine-tuning stage, where the cluster centroids are taken as model parameters to be updated via stochastic gradient descent. Before the fine-tuning stage, they all undergo a pretraining phase wherein the graph encoder and node representations are pretrained by minimizing an unsupervised loss, such as contrastive loss and the reconstruction loss. The differences among these methods exist in the design of the graph encoder and the design of additional modules. For instance, DAEGC [24] utilizes a graph attentional encoder to learn node representations. Both DFCN and DCRN adopt a complex graph encoder with three modules: GAE, IGAE, and the fusion module. Based on the complex encoder, DCRN designs a cluster- and node-level correlation reduction module.
- ***End-to-end methods.*** To verify the potential of diversifying the cluster centroids for improving graph clustering, we develop new versions of our DDGC and *Fine-tuning Methods* that omit the pretraining stage, which we refer to *End-to-end Methods*. These variants are denoted as DDGC *w/o pre*, DAEGC *w/o pre*, DFCN *w/o pre*, and DCRN *w/o pre*, respectively, which are initialized randomly and trained from scratch. We use an abbreviation of "*w/o pre*" to indicate removing the pretraining process from the training process. Moreover, DAGC [47], a deep attention-guided graph clustering with dual self-supervision, can be effectively trained in an end-to-end manner due to the proposed soft and hard self-supervision strategy. AGCC [46] is an end-to-end parallelly adaptive graph convolutional clustering model with two pathway networks to update the graph structure and extract the latent data features.

4.2. Experiment setup

We follow the experimental set-up of the baseline, DAEGC [24]), for a fair comparison, including the weight decay of 5×10^{-3}, Adam optimizer, the 1-th layer with 256 dimensions and 2-th layer with 16 dimensions in graph convolution encoder, the 30 pretraining epochs and the 200 fine-tuning epochs; While additional hyperparameters are selected according to the model performance, especially that introduced by our DDGC, including a, b and c in Eq. (17), as well as the epochs T for training the inner loop in Algorithm 1. The sensitivity analysis of c and the α controlling the diversity regularization term in Eq. (4b) have been provided in Fig. 12. We set T, a, and b to 300, 1.5, and 0.0005, which are selected according to the model performance and their validation procedures are provided in Fig. 16 in Appendix C.2. We set h to $\sqrt{\frac{\mathbb{MED}}{2\log N}}$, where $\mathbb{MED}$ is the median of the pairwise Euclidean distance between m cluster centroids $\{\boldsymbol{\mu}_i\}_{i=1}^m$, *i.e.*, $\mathbb{MED}(D)$ with $D = \{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2 \mid 0 \le i,j \le m\}$. Sensitivity analysis of h can be found in Appendix C.1. K' is set to 0 in DDGC *w/o pre*. For our DDGC, we followed DAEGC [24] for data pre-processing and testing. During the training of the seeking diverse bellwethers module, only the centroids $\{\boldsymbol{\mu}_i\}_{i=1}^m$ are learned, while the covariance matrix $\boldsymbol{\Sigma}_i$ of the underlying Gaussian mixtures is set as the identity matrix for computational convenience. For DCRN [27], DAEGC [24], and DFCN [26]

[3] https://dblp.uni-trier.de
[4] http://dl.acm.org/

Table 2
Performance comparison on graph datasets w.r.t. ACC (%), NMI (%) and ARI (%). The bold values indicate the best results. "*w/o pre*" is an abbreviation for "without pretraining".

Datasets		Cora			Citeseer			DBLP			ACM			AMAP		
Methods		ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
Two-step Methods	TADW	53.60$_{\pm 0.6}$	36.60$_{\pm 0.51}$	24.00$_{\pm 0.52}$	52.90$_{\pm 0.89}$	32.00$_{\pm 0.7}$	28.60$_{\pm 0.8}$	54.24$_{\pm 1.8}$	22.76$_{\pm 1.5}$	19.51$_{\pm 2.0}$	78.36$_{\pm 1.95}$	49.05$_{\pm 2.04}$	51.90$_{\pm 2.35}$	63.42$_{\pm 3.05}$	51.06$_{\pm 2.9}$	39.72$_{\pm 2.5}$
	GAE	38.90$_{\pm 0.51}$	17.82$_{\pm 0.35}$	14.79$_{\pm 0.46}$	38.00$_{\pm 0.8}$	17.40$_{\pm 0.7}$	14.10$_{\pm 1.2}$	61.21$_{\pm 1.2}$	30.80$_{\pm 0.9}$	22.02$_{\pm 1.4}$	84.52$_{\pm 1.4}$	55.38$_{\pm 1.9}$	59.46$_{\pm 3.1}$	71.57$_{\pm 2.48}$	62.13$_{\pm 2.79}$	48.82$_{\pm 4.57}$
	VGAE	38.90$_{\pm 0.8}$	16.00$_{\pm 0.5}$	11.32$_{\pm 0.2}$	39.20$_{\pm 0.4}$	16.30$_{\pm 0.3}$	10.10$_{\pm 0.5}$	58.59$_{\pm 0.1}$	26.92$_{\pm 0.1}$	17.92$_{\pm 0.1}$	84.13$_{\pm 0.2}$	53.20$_{\pm 0.5}$	57.72$_{\pm 0.7}$	68.35$_{\pm 0.9}$	56.93$_{\pm 0.82}$	42.18$_{\pm 0.6}$
	ARGE	64.00$_{\pm 1.0}$	44.90$_{\pm 0.8}$	35.20$_{\pm 0.68}$	57.30$_{\pm 1.2}$	35.00$_{\pm 0.72}$	34.10$_{\pm 0.6}$	59.22$_{\pm 1.0}$	23.16$_{\pm 0.5}$	22.35$_{\pm 0.45}$	86.29$_{\pm 2.36}$	56.21$_{\pm 1.98}$	63.37$_{\pm 1.5}$	69.28$_{\pm 1.7}$	58.36$_{\pm 1.43}$	44.18$_{\pm 0.88}$
Fine-tuning Methods	SDCN	60.01$_{\pm 0.40}$	40.13$_{\pm 0.39}$	33.32$_{\pm 0.27}$	66.79$_{\pm 0.3}$	40.91$_{\pm 0.3}$	41.05$_{\pm 0.4}$	68.05$_{\pm 1.8}$	39.50$_{\pm 1.3}$	39.15$_{\pm 2.0}$	89.45$_{\pm 0.2}$	68.31$_{\pm 0.3}$	72.91$_{\pm 0.4}$	53.44$_{\pm 0.81}$	44.85$_{\pm 0.83}$	31.21$_{\pm 1.23}$
	DAEGC	69.37$_{\pm 1.0}$	52.66$_{\pm 0.73}$	46.28$_{\pm 0.47}$	67.34$_{\pm 1.4}$	42.00$_{\pm 0.9}$	42.92$_{\pm 1.2}$	62.05$_{\pm 0.5}$	32.49$_{\pm 0.5}$	21.03$_{\pm 0.5}$	86.94$_{\pm 2.8}$	56.18$_{\pm 4.2}$	59.35$_{\pm 3.9}$	76.44$_{\pm 0.01}$	65.57$_{\pm 0.03}$	59.39$_{\pm 0.02}$
	AGCN	67.94$_{\pm 0.9}$	50.74$_{\pm 0.78}$	45.01$_{\pm 0.5}$	68.79$_{\pm 1.0}$	41.54$_{\pm 0.7}$	43.79$_{\pm 0.74}$	73.26$_{\pm 1.2}$	39.68$_{\pm 0.5}$	42.49$_{\pm 0.63}$	90.59$_{\pm 1.0}$	68.38$_{\pm 0.66}$	74.20$_{\pm 0.85}$	75.89$_{\pm 0.9}$	68.62$_{\pm 0.7}$	57.56$_{\pm 0.67}$
	DFCN	68.95$_{\pm 0.81}$	51.34$_{\pm 0.41}$	45.54$_{\pm 0.48}$	69.50$_{\pm 0.2}$	43.90$_{\pm 0.2}$	45.50$_{\pm 0.3}$	76.00$_{\pm 0.8}$	43.70$_{\pm 1.0}$	47.00$_{\pm 1.5}$	90.60$_{\pm 0.2}$	**72.90**$_{\pm 0.4}$	69.40$_{\pm 0.4}$	76.88$_{\pm 0.8}$	69.21$_{\pm 1.0}$	58.98$_{\pm 0.84}$
	DCRN	69.39$_{\pm 0.60}$	52.96$_{\pm 0.35}$	46.45$_{\pm 0.49}$	70.76$_{\pm 0.18}$	45.04$_{\pm 0.35}$	45.72$_{\pm 0.30}$	78.03$_{\pm 0.25}$	47.45$_{\pm 0.44}$	52.64$_{\pm 0.46}$	90.76$_{\pm 0.2}$	68.27$_{\pm 0.61}$	75.45$_{\pm 0.52}$	79.91$_{\pm 0.13}$	**73.64**$_{\pm 0.24}$	61.35$_{\pm 0.20}$
	DDGC(ours)	**72.14**$_{\pm 0.3}$	**54.10**$_{\pm 0.22}$	**49.90**$_{\pm 0.19}$	**71.83**$_{\pm 0.3}$	**45.47**$_{\pm 0.23}$	**47.52**$_{\pm 0.2}$	**79.47**$_{\pm 0.1}$	**48.37**$_{\pm 0.13}$	**54.50**$_{\pm 0.09}$	**91.64**$_{\pm 0.12}$	71.31$_{\pm 0.09}$	**76.86**$_{\pm 0.1}$	**81.02**$_{\pm 0.02}$	72.18$_{\pm 0.02}$	**64.82**$_{\pm 0.01}$
End-to-end Methods	AGCC	67.56$_{\pm 3.1}$	49.83$_{\pm 2.5}$	44.37$_{\pm 1.82}$	68.08$_{\pm 3.9}$	40.86$_{\pm 2.4}$	41.82$_{\pm 2.2}$	73.45$_{\pm 3.7}$	40.36$_{\pm 2.65}$	44.40$_{\pm 2.1}$	90.38$_{\pm 4.12}$	68.34$_{\pm 3.4}$	73.73$_{\pm 3.98}$	78.82$_{\pm 4.77}$	72.85$_{\pm 3.48}$	60.81$_{\pm 3.23}$
	DAGC	68.76$_{\pm 2.7}$	50.48$_{\pm 2.1}$	44.91$_{\pm 1.5}$	69.43$_{\pm 3.21}$	43.68$_{\pm 2.76}$	45.06$_{\pm 2.53}$	77.82$_{\pm 4.10}$	46.74$_{\pm 3.62}$	52.34$_{\pm 3.1}$	90.74$_{\pm 4.0}$	69.39$_{\pm 2.9}$	74.66$_{\pm 3.2}$	78.43$_{\pm 3.6}$	72.23$_{\pm 3.52}$	61.37$_{\pm 3.2}$
	DAEGC *w/o pre*	54.65$_{\pm 8.5}$	43.88$_{\pm 7.5}$	35.34$_{\pm 4.74}$	46.93$_{\pm 5.72}$	27.07$_{\pm 4.36}$	23.53$_{\pm 3.98}$	54.55$_{\pm 7.36}$	24.60$_{\pm 4.0}$	24.87$_{\pm 3.2}$	87.93$_{\pm 10.2}$	62.04$_{\pm 8.3}$	67.52$_{\pm 6.8}$	56.78$_{\pm 6.3}$	55.92$_{\pm 5.28}$	42.62$_{\pm 4.32}$
	DFCN *w/o pre*	53.36$_{\pm 5.2}$	38.33$_{\pm 3.76}$	28.79$_{\pm 3.28}$	45.33$_{\pm 5.89}$	26.90$_{\pm 2.52}$	23.56$_{\pm 2.0}$	43.31$_{\pm 6.05}$	11.04$_{\pm 2.1}$	9.11$_{\pm 1.9}$	89.85$_{\pm 8.3}$	67.25$_{\pm 5.9}$	72.54$_{\pm 8.4}$	58.90$_{\pm 6.2}$	53.27$_{\pm 5.7}$	37.98$_{\pm 3.0}$
	DCRN *w/o pre*	50.92$_{\pm 6.9}$	33.28$_{\pm 4.58}$	22.90$_{\pm 3.9}$	57.80$_{\pm 7.3}$	29.20$_{\pm 2.5}$	28.35$_{\pm 2.1}$	46.14$_{\pm 3.2}$	16.18$_{\pm 1.5}$	13.16$_{\pm 1.6}$	88.94$_{\pm 4.9}$	64.16$_{\pm 3.2}$	69.96$_{\pm 2.92}$	58.12$_{\pm 3.15}$	54.32$_{\pm 2.89}$	43.09$_{\pm 3.0}$
	DDGC *w/o pre*(ours)	**69.32**$_{\pm 4.0}$	**52.74**$_{\pm 3.1}$	**46.37**$_{\pm 2.5}$	**70.18**$_{\pm 3.8}$	**44.34**$_{\pm 1.91}$	**45.46**$_{\pm 2.0}$	**78.36**$_{\pm 3.1}$	**47.75**$_{\pm 2.7}$	**53.31**$_{\pm 2.6}$	**91.37**$_{\pm 2.98}$	**70.56**$_{\pm 2.42}$	**76.20**$_{\pm 2.0}$	**80.33**$_{\pm 2.62}$	**70.91**$_{\pm 2.1}$	**63.77**$_{\pm 1.91}$

on Cora, we executed their official source code following the settings provided in their original literature and reported their average results. DDGC *w/o pre*, DAEGC *w/o pre*, DFCN *w/o pre*, and DCRN *w/o pre* are initialized randomly and trained from scratch. For DFCN [26] on other datasets, we directly reported the corresponding values as listed in DCRN [27].

Metrics. Clustering results are obtained by assigning nodes to the cluster corresponding to the maximum value in the predicted assignment distribution vector. We adopt the widely-used metrics to evaluate the clustering performance [14], Accuracy (ACC), Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI). Higher values for all metrics indicate better clustering results. The best mapping between the predicted cluster-ID and the given class-ID was constructed using the Kuhn-Munkres [48].

4.3. Overall performance

To evaluate the clustering performance of the proposed DDGC, we conduct extensive experiments on five benchmark datasets, comparing DDGC against two categories comprising 11 baselines, including *two-step*, and *Fine-tuning Methods*. Moreover, to verify the potential of diversifying centroids and sharpening assignment distribution, we compare DDGC *w/o pre* with *End-to-end methods*, *i.e.*, our adapted versions of baselines without pretraining, namely DAEGC *w/o pre*, DFCN *w/o pre*, and DCRN *w/o pre*. The comprehensive clustering performance is summarized in Table 2. We have the following observations:

- DDGC achieves state-of-the-art clustering performance on all datasets, which outperforms different methods, including the *two-step methods*, and *Fine-tuning Methods* in terms of ACC, NMI, and ARI metrics. Among them, DFCN [26] and DCRN [27] are considered as the strongest deep clustering models. Specifically, compared to *Two-step Methods* including TADW [44], GAE [28], VGAE [28], and ARGE [19], DDGC consistently outperforms them by a considerable margin. We conjecture that lacking the mutual interaction of embedding learning and clustering training, *Two-step Methods* may suffer from the mismatch between two objectives and thus obtain inferior clustering performance compared to *Fine-tuning Methods* and DDGC. Moreover, our DDGC exhibits superior performance over *Fine-tuning Methods* due to incorporating mechanisms including diversifying cluster centroids in the SDB module and the sharper assignment distribution in the SCA module. Generally, the *Fine-tuning Methods* outperform the *Two-step Methods* significantly, benefiting from the joint training of the cluster centroids and the node representations.
- DDGC *w/o pre* demonstrates significant superiority over the existing *End-to-end methods*, including AGCC and DAGC, and all developed *End-to-end methods*, including DAEGC *w/o pre*, DFCN *w/o pre*, and DCRN *w/o pre*. Specifically, DDGC *w/o pre* significantly improves the best accuracy by 19.92%, 13.87%, 24.92%, 3.71%, and 21.52% on Cora, Citeseer, DBLP, ACM, and AMAP, respectively. The mechanisms of diversity- and compactness-enhanced clustering not only elevate the clustering performance but also facilitate effective end-to-end training of DDGC. Besides, DFCN *w/o pre* and DCRN *w/o pre* exhibit notably inferior performance compared to DAEGC *w/o pre* on Cora, DBLP, and ACM. That is because the complex graph encoder adopted by DFCN [26] and DCRN [27] includes three modules comprising GAE, IGAE, and its fusion module to learn node representations, which is hard to be trained from scratch together with the cluster centroids. That means *DFCN and DCRN are unsuitable for training from scratch.*
- The following evidence can verify that DDGC has significantly reduced the reliance on pre-training: (i) when removing the pre-training stage, the clustering performance (ACC, NMI, ARI) of baselines, *i.e.*, DAEGC *w/o pre*, DFCN *w/o pre*, DCRN *w/o pre*, has significantly decreased. However, the clustering performance of DDGC *w/o pre* is very close to those of our DDGC (with pre-training); (ii) compared with baselines including DAEGC, DFCN, and DCRN, our DDGC *w/o pre*, even skipping the pre-training phase, still achieves comparable clustering performance on Cora and Citeseer datasets, and superior performance on DBLP, ACM and AMAP datasets; (iii) compared with all baselines without pre-training (DAEGC *w/o pre*, DFCN *w/o pre*, DCRN *w/o pre*), DDGC *w/o pre* significantly outperform their clustering performance, in terms of all metrics, *i.e.*, ACC, NMI, ARI. More experiments are provided in Appendix E.

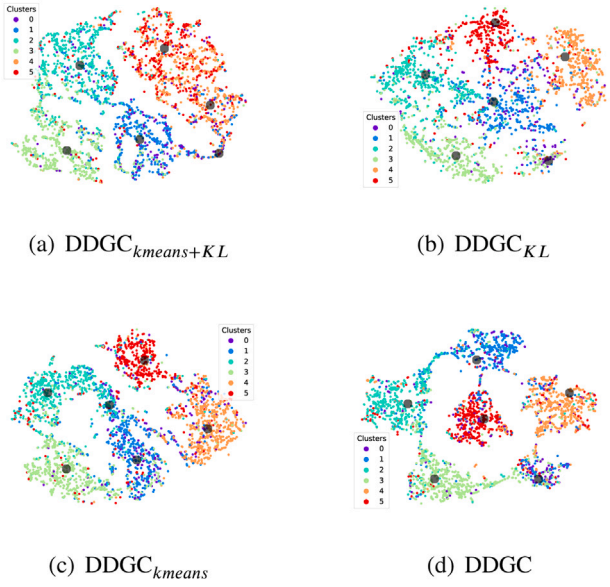
4.4. Ablation analysis

To further investigate effectiveness of each component in DDGC, we develop three distinct variants of DDGC: DDGC$_{KL}$, DDGC$_{kmeans}$, and DDGC$_{kmeans+KL}$ by substituting the proposed Minimum Entropy (ME)-based clustering loss with the conventional KL-based loss, substituting the seeking-diverse-bellwethers (SDB) module with the classical kmeans algorithm, and employing both the KL-based loss and kmeans, respectively. We conduct the ablation study by comparing DDGC with these three variants on five datasets, and their clustering results are presented in Table 3.

As observed, we can demonstrate several key insights: (1) DDGC consistently outperforms DDGC$_{KL}$ with an average accuracy margin of 1.2% across all datasets, affirming the effectiveness of our ME-based clustering loss in enhancing the clustering performance; (2) Compared with DDGC$_{kmeans}$, DDGC achieves an average accuracy gain of 1.8% across all dataset, which demonstrates the effectiveness of the diversifying cluster centroids in the seeking-diverse-bellwethers (SDB)

Table 3
Ablation study w.r.t. ACC (%), NMI (%) and ARI (%).

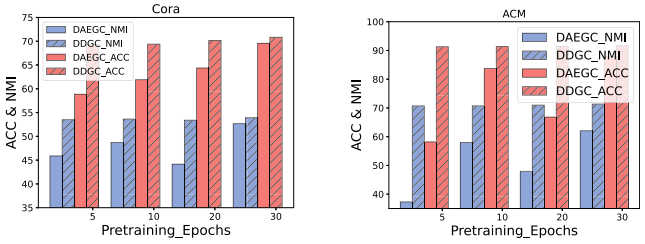
Methods	$DDGC_{kmeans+KL}$			$DDGC_{KL}$			$DDGC_{kmeans}$			DDGC		
Datasets	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
Cora	67.76	49.24	45.64	69.76	49.17	45.66	68.87	50.32	45.43	**72.14**	**54.10**	**49.90**
Citeseer	69.70	43.20	45.26	70.06	45.22	43.73	70.09	43.42	45.76	**71.83**	**45.47**	**47.52**
DBLP	75.06	43.14	45.68	75.67	43.30	46.72	77.57	46.99	47.86	**79.47**	**48.37**	**54.50**
ACM	86.08	60.69	64.49	88.56	64.71	69.52	90.71	69.72	74.66	**91.64**	**71.31**	**76.86**
AMAP	75.23	58.76	53.49	79.45	65.46	61.40	77.25	65.81	56.55	**81.02**	**72.18**	**64.82**



(a) $DDGC_{kmeans+KL}$ (b) $DDGC_{KL}$

(c) $DDGC_{kmeans}$ (d) DDGC

Fig. 9. Clustering visualization for the Citeseer dataset. The grey dots denote the learned centroid of each cluster.



(a) Cora (b) ACM

Fig. 10. Comparison of the baseline DAEGC and DDGC performance (ACC (%), NMI (%)) with different pretraining epochs on Cora and ACM datasets. Best viewed in color.

module; (3) The SDB module exhibits a more pronounced impact on the clustering results compared to the ME-based clustering loss. This is evidenced by $DDGC_{KL}$ outperforming $DDGC_{kmeans}$ across all datasets in terms of ACC, NMI, and ARI metrics; (4) DDGC demonstrates a larger accuracy increase, averaging 3.5%, over five datasets compared to $DDGC_{kmeans+KL}$. This suggests that simultaneously incorporating the SDB module and ME-based clustering loss into our framework further enhances clustering performance by promoting the compactness and diversity of clusters. Furthermore, based on the baseline, DAEGC, we develop other variants to verify the effectiveness of the SDB module and the ME-based clustering loss, such as $DAEGC_{ME}$ that replaces the KL-based loss with our ME loss. The results of these variants are presented in Appendix.

4.5. Cluster visualization

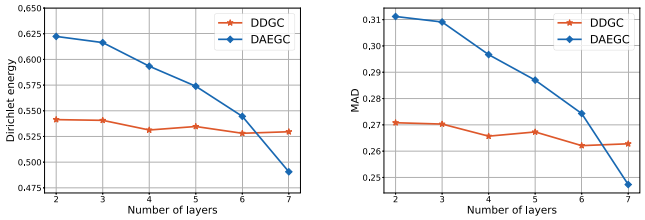
To visually assess the effect of the diverse cluster centroids and the sharpening assignment distribution in DDGC, we conduct t-SNE visualization for the above three variants, including $DDGC_{kmeans+KL}$, $DDGC_{KL}$, and $DDGC_{kmeans}$ on the Citeseer dataset. Fig. 9 visualizes the node representations learned by DDGC and these three variants, which are projected into a two-dimensional space using the t-SNE algorithm [49]. As indicated in Table 1, the Citeseer dataset comprises 6 clusters. Different colors represent the classes of nodes, and the black dots denote the cluster centroids.

The visualized clustering results reveal several key findings: (1) In Fig. 9(b) and (d), the cluster centroids, learned by $DDGC_{KL}$ and DDGC, appear as pentagrams, evenly distributed in space, but the cluster centroids in Fig. 9(a) and (c) do not exhibit the nonuniform pattern. This is the evidence that the diversity regularizer in the gradient function, Eq. (4b), effectively enables the SDB module to learn diverse cluster centroids; (2) Fig. 9(c) and (d) demonstrate that the clusters learned by both $DDGC_{kmeans}$ and DDGC are more compact compared to $DDGC_{kmeans+KL}$ and $DDGC_{KL}$. This compactness arises because the ME-based clustering loss facilitates the clustering assignment distribution of each node to approach a one-hot distribution, resulting in more compact clusters than those achieved with the conventional KL-based loss; (3) Notably, DDGC in Fig. 9(d) presents well-separated, diverse and compact clusters compared to $DDGC_{kmeans+KL}$ in Fig. 9(a). Its improvement can be attributed to the seeking diverse bellwether module and the sharpening assignment mechanism in DDGC. These components enable DDGC to learn diversified centroids and compact clusters, leading to learning discriminative node representations and stabilizing the training process.

4.6. Analysis of stability to pretraining

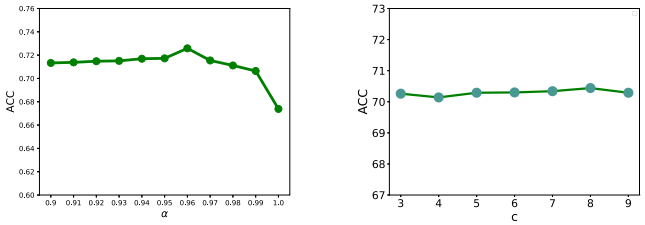
To further demonstrate the stability of DDGC with respect to pretraining, we compared it with the baseline DAEGC [24], as they utilize the same attentional autoencoder. We vary the number of pretraining epochs within $\{5, 10, 20, 30\}$ on Cora and report the fine-tuning results in Fig. 10. It is evident from the results that our DDGC consistently outperforms DAEGC across all pretraining epochs and mitigates the dependence on the well-pretrained encoder.

From the observations depicted in Fig. 10, we can draw the following conclusions: (1) The baseline DAEGC demonstrates high sensitivity to the quality of the pretraining encoder, as its clustering performance significantly decreases when the number of pretraining epochs is reduced on Cora and ACM datasets. That is because the minimize the KL-based clustering loss and the reconstruction loss to simultaneously train the cluster centroids and the graph encoder, where the centroids are taken as the parameters, limiting the exploration of representation space; (2) In contrast, our method DDGC performs stably and robustly over accuracy and NMI, regardless of the variations in the pretraining encoder. DDGC consistently achieves an accuracy of approximately 71% and an NMI of 54% on Cora, as well as an accuracy of approximately 91.5% and an NMI of 71.5% on the ACM dataset across all pretraining epochs. The experimental results confirm the effectiveness of the diverse centroids and sharpening assignment mechanism. Our framework of DDGC decomposes the optimization of centroids and graph encoder, leading to better clustering performance.



(a) Dirichlet energy (b) MAD

Fig. 11. Trained our DDGC and DAEGC on Cora dataset using the same pre-trained GAE encoder of 30 epochs, showing two different measures for increasing number of layers ranging from 2 to 7: (**left**) Dirichlet energy of the layer-wise node features, (**right**) *Mean Average Distance* (MAD).



(a) ACC (%) vs. α (b) ACC (%) vs. c

Fig. 12. Clustering accuracy (%) vs. hyperparameters α and c on Cora dataset.

4.7. Studies on alleviating the over-smoothing problem

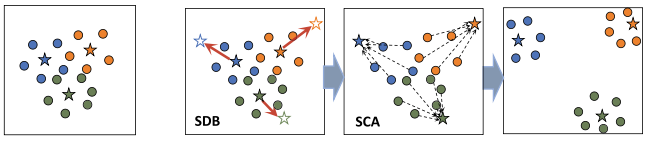
We conduct experiments to validate that our DDGC model can alleviate the *over-smoothing* issue to some extent. The *over-smoothing* problem refers to the significant deterioration in clustering performance as the number of layers in GNNs increases. Its occurrence can be verified in Fig. 14. We compare two *over-smoothing* metrics, *i.e.*, *Dirichlet energy* and *Mean Average Distance* (MAD) (we introduce them in Appendix B.2), of our DDGC and the baseline DAEGC [24] by increasing the depth of graph attentional autoencoder from 2 to 7 layers.

Fig. 11 displays these two metrics of DAEGC [24] and DDGC as the number of layers in graph encoder increases. The experimental findings highlight the following insights: (1) DAEGC exhibits a gradual decline in *Dirichlet energy* and MAD as the number of layers increases, indicating suffering from *over-smoothing*; (2) In contrast to that, DDGC alleviates the issue of *over-smoothing* by keeping the layer-wise *Dirichlet energy* and MAD approximately constant, showcasing its ability to learn discriminative node representations by separating and condensing clusters.

4.8. Hyperparameter analysis

We investigate the influence of hyperparameters α in Eq. (4b) and the amplitude c in Eq. (17). The α plays a pivotal role in balancing the term of fitting node representations and the diversity regularizer for diversifying the cluster centroids. The amplitude c is more important to influence the value of the balance factor η, where a and b are set to 1.5 and 0.0005, respectively. We conduct experiments to show the effect of these two parameters on Cora datasets.

From the results illustrated in Fig. 12, it can be observed that DDGC is insensitive to hyperparameter c. The accuracy metric has slight increments with the increasing value c. Notably, the clustering performance of our DDGC fluctuates regarding the hyperparameter α with DDGC achieving its highest accuracy when α is set to 0.96.



(a) Centroids (b) Bellwethers

Fig. 13. Comparison of centroids and bellwethers. Three colored circles represent nodes of three clusters. The red arrow indicates the diversifying centroids of SDB. The dashed arrow denotes the bellwether role of centroids in SCA, which pulls intra-cluster nodes to move toward their own centroids.

5. Conclusion

Most existing deep graph clustering models struggle to achieve optimal performance due to their cluster-friendly representation learning and insufficient support for learning diversified clusters. To address their limitations, we proposed the Diversity-promoting Deep Graph Clustering framework (DDGC). DDGC embodies two crucial principles in clustering: minimizing the intra-cluster variance and maximizing the inter-cluster variance. We introduce a diversity term into the process of centroids learning, enabling DDGC to learn diverse cluster centroids and enhance node discrimination. Additionally, our proposed sharpening clustering assignment mechanism embedded in our ME-based clustering loss further consolidates clusters. These techniques enable stable training of DDGC from scratch but also significantly improve clustering performance. DDGC exhibits remarkable improvement in performance and robustness compared to state-of-the-art models.

CRediT authorship contribution statement

Peiyao Zhao: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **Xin Li:** Writing – review, Supervision, Software, Funding acquisition, Formal analysis . **Yuangang Pan:** Writing – review, Supervision, Methodology, Formal analysis. **Ivor W. Tsang:** Supervision, Resources, Funding acquisition, Formal analysis. **Mingzhong Wang:** Writing – review & editing. **Lejian Liao:** Supervision.

Declaration of competing interest

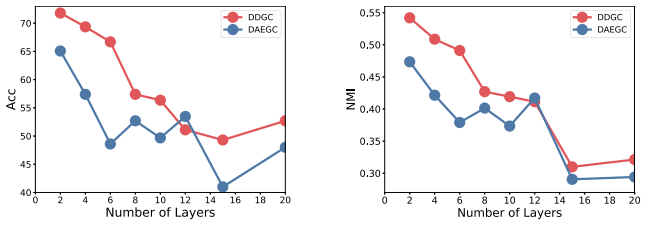
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would also like to thank the anonymous reviewers for their constructive feedback, which greatly helped us to improve the quality and clarity of the paper. This work was partially supported by the National Natural Science Foundation of China (NSFC) under Grants 92270125 and 62276024.

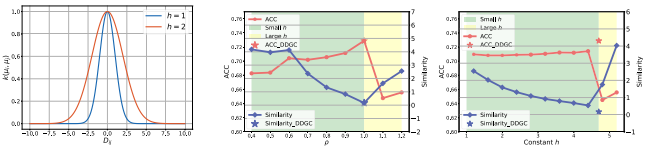
Appendix A. Difference between centroids and bellwethers

We clarify two meanings of "bellwethers": (i) the means of node features within clusters, (ii) leading the update of latent node features within clusters, facilitated by a diversity regularization term in Eq. (4b). Finally, we can obtain well-separate clusters with wider cluster boundaries. Fig. 13(b) illustrates the dynamic process of our "bellwethers" leading representation learning. The traditional "centroids" are only the centers of clusters, corresponding to the first meaning, shown in Fig. 13(a).



(a) ACC (%) (b) NMI

Fig. 14. Comparison of DAEGC and DDGC performance (ACC (%), NMI) with varying the depth of graph encoder on Cora dataset. DDGC boosts overall performance by facilitating increased diversity with the addition of layers. Best viewed in color.



(a) $h = 1$ vs. $h = 2$ (b) $\rho \cdot \mathbb{MED}$ (c) Fixed h

Fig. 15. Comparison of different bandwidths of our RBF kernel $k(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) = \exp(-\frac{D_{ij}^2}{2h^2})$ with $D_{ij} = \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|$. **(a)** $h = 1$ vs. $h = 2$. **(b) Changed** h **across iterations**: the accuracy (ACC) performance and the centroid similarity of DDGC with increasing ρ from 0.4 to 1.2 (we use $\rho \cdot \mathbb{MED}(D)$ replace $\mathbb{MED}(D)$ in h). **(c) Constant** h **across iterations**: the ACC performance with increasing constant h from 1.2 to 5.2. In Fig. 15(b) and (c), the green and yellow shadows respectively denote the **small** and **large** h. The red and blue **Stars** denote the clustering results of our h: for Fig. 15(b), $\rho = 1$; for Fig. 15(c), our h is averaged approximately at 4.7 over all iterations.

Appendix B. Over-smoothing

B.1. Over-smoothing on graph clustering

Some **graph smoothing improves performance before oversmoothing occurs, while** ***too much*** **smoothing inevitably leads to** ***oversmoothing*** [50]. *Oversmoothing* hence makes the node features lose information for distinguishing nodes quickly when the layer size goes to infinity. The goal of graph clustering is to partition nodes into distinct clusters, ensuring that nodes within clusters are more similar than those between clusters. The reduction in node discrimination caused by the *over-smoothing* problem results in the DAEGC and DDGC models producing clustering-unfriendly representations, which hinder the ability to distinguish groups in the clustering task, thereby significantly impairing model performance. Fig. 14 illustrates that DDGC and DAEGC suffered from the *over-smoothing* issue.

B.2. Dirichlet energy and mean average distance

Besides, to further verify the mitigating *over-smoothing* of our DDGC, we introduce two additional metrics, *Dirichlet energy* and Mean Average Distance. Existing approaches [51–53] to measure over-smoothing in deep GNNs have mainly been based on the concept of *Dirichlet energy* on graphs,

$$D(\mathbf{Z}^n) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} \|\mathbf{Z}_i^n - \mathbf{Z}_i^n\|_2^2 \tag{18}$$

where $\mathbf{Z}^n$ is the learned node embeddings of all nodes at the nth layer, and $\mathcal{N}_i$ denotes the neighbors set of the node i. Besides, *Mean Average Distance* (MAD) is also an additional evaluation metric to measure over-smoothing broadly used in a variety of existing approaches [54,55]:

$$\mathcal{M}(\mathbf{Z}^n) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} (1 - \frac{\mathbf{Z}_i^{nT} \mathbf{Z}_j^n}{\|\mathbf{Z}_i^n\| \|\mathbf{Z}_j^n\|}) \tag{19}$$

Both metrics measure the distance in representations between nodes and their neighbors. We conduct the following experiments to compare *Dirichlet energy* and MAD of our DDGC and the baseline with increasing layers. Fig. 11 in Section 4.7 has demonstrated that our DDGC indeed mitigates the over-smoothing problem compared to DAEGC.

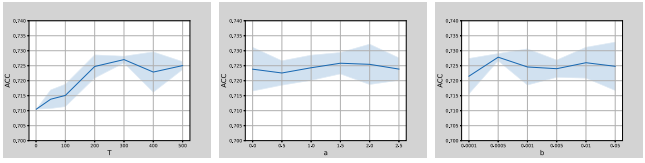
Appendix C. Analysis of hyperparameters

C.1. Analysis of h in RBF kernel

The institution of h is that it can normalize the Euclidean distance such that the sum of the kernel values of a query centroid $\boldsymbol{\mu}_i$ is 1, *i.e.*, $\sum_{j=1}^{m} k(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) \approx N \exp(-\frac{1}{h^2}\mathbb{MED}^2) = 1$, where $k(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) = \exp(-\frac{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2}{2h^2})$ is our adopted RBF kernel. That can ensure that our kernel is normalized and ranges from 0 (in the infinite-distance limit) to 1 (when $\boldsymbol{\mu}_i = \boldsymbol{\mu}_j$). Note that in this way, the bandwidth h actually changes adaptively across the iterations. Fig. 15(a) illustrates the difference of kernel functions with different h sizes. *The right value of* h *is important to decide which centroids should be considered similar*. Too-large h makes the influence of the kernel span a larger area, making centroids far apart assigned the non-negligible or high similarity. Too-small h makes the influence of the kerne more localized around the query centroid $\boldsymbol{\mu}_i$, assigning higher similarity only to close centroids. In the following, we will conduct experiments to analyze the effect of bandwidth h on model performance.

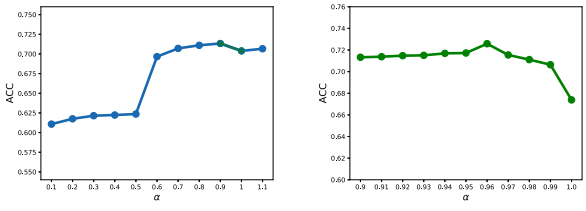
Influence on model performance. To analyze the influence of h on our DDGC method, we develop two types of h as variants: with epoch increasing (i) adaptively changing and (ii) remaining constant h. For (i), we replace $\mathbb{MED}$ of our h by $\rho \cdot \mathbb{MED}$, *i.e.*, $\sqrt{\frac{\rho \cdot \mathbb{MED}}{2 \log N}}$, with $\rho \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.1, 1.2\}$. The h variants of $\rho > 1$ (resp. $\rho < 1$) are **large** (resp. **small**) h. For (ii), we pre-fix constant h with $h \in \{1.2, 1.6, 2.0, 2.4, 2.8, 3.2, 3.6, 4.0, 4.4, 4.8, 5.2\}$, where the variants are considered as **large** (resp. **small]**) h if $h > 4.7$ (resp. $h < 4.7$) (because the h of our $\mathbb{MED}$ is averaged at 4.7 across all iterations). We evaluate our h and those variants by the clustering accuracy (ACC) performance and the metric of centroid diversity (or similarity) of our paper.

From Fig. 15, we can summarize the following conclusions: (i) For Fig. 15(a), when $\rho < 1$, as ρ increases the ACC performance of $\rho \cdot \mathbb{MED}$ has been improved steadily due to the decreasing similarity among centroids. Conversely, when $\rho > 1$, with increasing ρ the ACC has been decreasing due to the increasing similarity among centroids. Fig. 15 exhibits a similar pattern. (ii) In both Fig. 15(b) and (c), our h with $\mathbb{MED}$ has achieved better accuracy (ACC) (**red Stars**), compared to other h variants, including $\rho \cdot \mathbb{MED}$ and fixed constants, which has verified that too-large or small bandwidth h can adversely affect the model clustering performance. That is because our h in our kernel has made DDGC learn more diverse cluster centroids (**blue Stars**) with fewer similarities than other variants h.



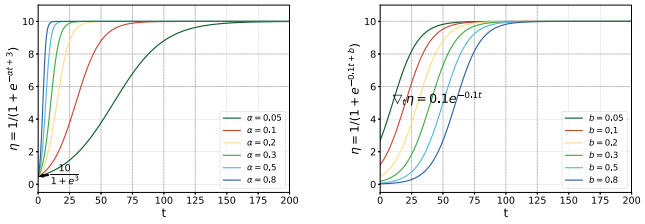
(a) T (b) a (c) b

Fig. 16. The clustering accuracy (%) of DDGC with different T, a and b.



(a) (b)

Fig. 17. Left: Selection process for the optimal sub-range within the broader range $[0.1, 1.1]$ for hyperparameter α; **Right:** Local search for the optimal value within the sub-range.



(a) a vs. η (b) b vs. η

Fig. 18. The comparison of η with different a and b. (a) Fixing b, the larger a is, the faster the weight of the entropy regularization increases and vice versa. (b) Fixing a, the larger b is, the larger the weight of the regularization is, and vice versa.

C.2. Analysis of T, a and b

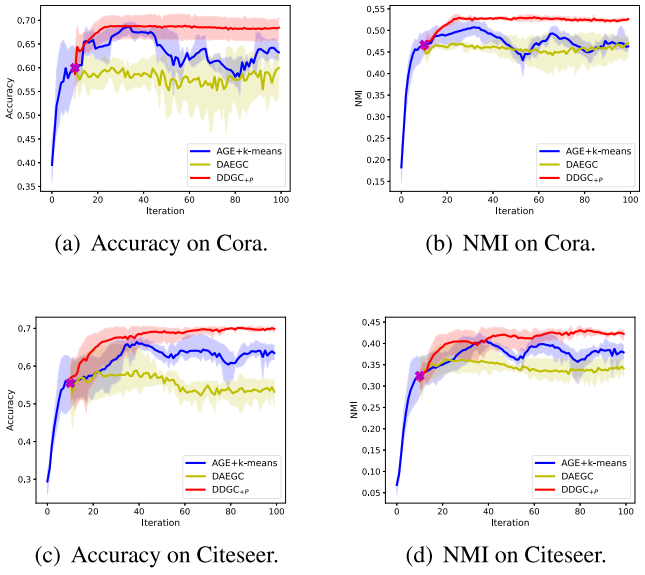
In hyperparameter settings, we set the T, a, and b to 300, 1.5, and 0.0005, which are selected by the following sensitivity analysis and validation procedures in Fig. 16.

C.3. Analysis of diversity coefficient α

We have demonstrated the process of selecting the range for hyperparameters a. Specifically, we first identified the optimal sub-range for a as $[0.9, 1]$ by searching within the broader range $[0.1, 1.1]$, as illustrated in Fig. 17(a). Then we locally searched in the sub-range $[0.92, 1]$ and found the optimal value as 0.96 in Fig. 17(b).

Appendix D. Analysis of dynamic coefficient

Specifically, c is the upper bound of the balance weight. Specifically, taking $c = 10$ as an example, Fig. 18(a) shows the curves of $\eta(t; a, b = 3) = \frac{10}{1+e^{-at+3}}$ with only varying a and fixing $b = 3$. We find that varying a only changes the rate at which the entropy regularization merges in the total loss, but the initial weight remains unchanged $\frac{10}{1+e^{3}}$. Fig. 18(b) shows the curves of $\eta(t; a, b = 3) = \frac{10}{1+e^{-0.1t+b}}$ with only varying b and fixing $a = 0.1$. We find that varying b only changes the initial weight, but the rate remains unchanged, *i.e.*, all curves have the same gradient $\nabla_t \eta = 0.1e^{-0.1t}$. That is why we call a the rate and b the initial weight.



(a) Accuracy on Cora. (b) NMI on Cora.

(c) Accuracy on Citeseer. (d) NMI on Citeseer.

Fig. 19. The performance of different methods w.r.t. iteration on Cora and Citeseer. Both DAEGC and DDGC use the same AGE parameters and centroids from the 10th iteration of the pretraining as the initialization settings. The pretraining results at 10th iteration are marked with a bold purple cross.

Appendix E. Additional analysis of stability

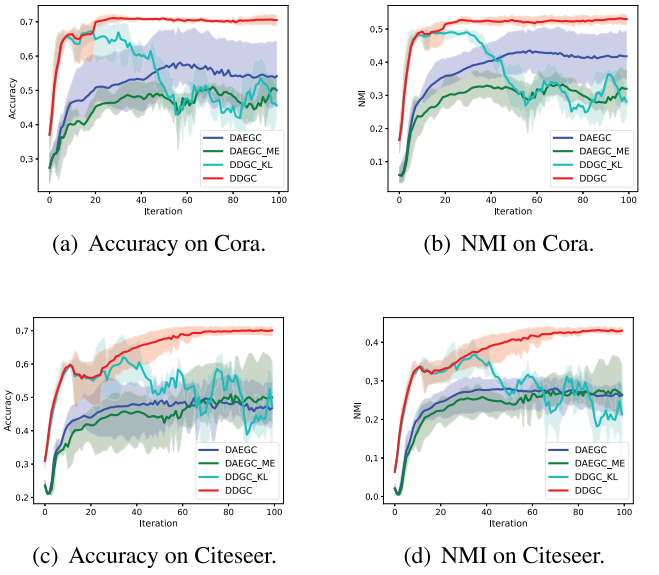
In Fig. 19, we visualize the performance (ACC and NMI) of our DDGC (refer to DDGC$_{+P}$ and DAEGC both initialized by the same pretrained encoder at 10 pretrained epoch in Fig. 19, where the bold purple cross is the training curve of pretraining autoencoder for 100 epoch. The results show that: (1) DAEGC yields inferior clustering performance, given the inadequately trained node representations. It only slightly improves the clustering performance produced by autoencoder at 10th epoch (labeled as a bold purple cross); (2) DDGC achieves a superior performance despite the sub-optimal initial node presentations. This is evident that DDGC can mitigate the reliance on the well-pretrained encoder.

Appendix F. Additional ablation analysis

To provide a more detailed analysis regarding the effectiveness of different components in different models, we developed DAEGC_ME, a variant of DAEGC, by replacing the KL divergence regularizer in DAEGC with the minimum entropy, and developed DDGC_KL by using KL divergence as the regularizer in DDGC. The results illustrated in Fig. 20 demonstrate that: (1) DDGC presents the most stable training curve and converges faster. It clearly performs better than DDGC_KL, which suffers severe fluctuations. We conclude that minimum entropy fits better in the DDGC framework than KL divergence, as DDGC is more likely to learn task-friendly representations for the graph clustering, which helps to avoid the model collapse, due to the use of a diversity centroids learning module. Therefore, DDGC performs better with a more confident clustering assignment mechanism. In comparison with KL divergence, minimum entropy provides a one-hot-like assignment, which contributes to stable learning. (2) DAEGC and DAEGC_ME present inferior performance than DDGC. However, KL divergence fits better in the DAEGC framework, as KL divergence offers a softer clustering assignment than minimum entropy, having more chances to adjust the optimization direction needed in DAEGC.

Data availability

Data will be made available on request.



(a) Accuracy on Cora. (b) NMI on Cora.

(c) Accuracy on Citeseer. (d) NMI on Citeseer.

Fig. 20. The performance of DAEGC and DDGC with KL divergence or entropy module w.r.t. iteration on Cora and Citeseer.

References

[1] Matthew B. Hastings, Community detection as an inference problem, Phys. Rev. E 74 (3) (2006) 035102.

[2] Di Jin, Zhizhi Yu, Pengfei Jiao, Shirui Pan, Dongxiao He, Jia Wu, S. Yu Philip, Weixiong Zhang, A survey of community detection approaches: From statistical modeling to deep learning, IEEE Trans. Knowl. Data Eng. 35 (2) (2021) 1149–1170.

[3] Qi Xuan, Jinhuan Wang, Minghao Zhao, Junkun Yuan, Chenbo Fu, Zhongyuan Ruan, Guanrong Chen, Subgraph networks with application to structural feature space expansion, IEEE Trans. Knowl. Data Eng. 33 (6) (2021) 2776–2789.

[4] Su-Yeon Kim, Tae-Soo Jung, Eui-Ho Suh, Hyun-Seok Hwang, Customer segmentation and strategy development based on customer lifetime value: A case study, Expert Syst. Appl. 31 (1) (2006) 101–107.

[5] Nairouz Mrabah, Mohamed Bouguessa, Mohamed Fawzi Touati, Riadh Ksantini, Rethinking graph auto-encoder models for attributed graph clustering, IEEE Trans. Knowl. Data Eng. 35 (9) (2023) 9037–9053.

[6] Youpeng Hu, Xunkai Li, Yujie Wang, Yixuan Wu, Yining Zhao, Chenggang Yan, Jian Yin, Yue Gao, Adaptive hypergraph auto-encoder for relational data clustering, IEEE Trans. Knowl. Data Eng. 35 (3) (2021) 2231–2242.

[7] Yunxiao Zhao, Liang Bai, Contrastive clustering with a graph consistency constraint, Pattern Recognit. 146 (2024) 110032.

[8] Yi Wen, Suyuan Liu, Xinhang Wan, Siwei Wang, Ke Liang, Xinwang Liu, Xihong Yang, Pei Zhang, Efficient multi-view graph clustering with local and global structure preservation, in: Abdulmotaleb El-Saddik, Tao Mei, Rita Cucchiara, Marco Bertini, Diana Patricia Tobon Vallejo, Pradeep K. Atrey, M. Shamim Hossain (Eds.), Proceedings of the 31st ACM International Conference on Multimedia, MM 2023, Ottawa, on, Canada, 29 October 2023- 3 November 2023, ACM, 2023, pp. 3021–3030.

[9] Ke Wang, Zanting Ye, Xiang Xie, Haidong Cui, Tao Chen, Banteng Liu, MLN-net: A multi-source medical image segmentation method for clustered microcalcifications using multiple layer normalization, Knowl.-Based Syst. 283 (2024) 111127.

[10] Feng Zhao, Zhilei Xiao, Hanqiang Liu, Zihan Tang, Jiulun Fan, A knee point driven kriging-assisted multi-objective robust fuzzy clustering algorithm for image segmentation, Knowl.-Based Syst. 271 (2023) 110522.

[11] Jean-Paul Ainam, Ke Qin, Jim Wilson Owusu, Guoming Lu, Unsupervised domain adaptation for person re-identification with iterative soft clustering, Knowl.-Based Syst. 212 (2021) 106644.

[12] Chunteng Bao, Diju Gao, Yi Ding, Lihong Xu, Erik D. Goodman, Many-task evolutionary algorithm with adaptive knowledge transfer via density-based clustering, Knowl.-Based Syst. 278 (2023) 110906.

[13] Qing Tian, Jixin Sun, Cluster-based dual-branch contrastive learning for unsupervised domain adaptation person re-identification, Knowl.-Based Syst. 280 (2023) 111026.

[14] Rongkai Xia, Yan Pan, Lei Du, Jian Yin, Robust multi-view spectral clustering via low-rank and sparse decomposition, in: Carla E. Brodley, Peter Stone (Eds.), Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, QuÉbec City, QuÉbec, Canada, AAAI Press, 2014, pp. 2149–2155.

[15] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, Tie-Yan Liu, Learning deep representations for graph clustering, in: Carla E. Brodley, Peter Stone (Eds.), Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, QuÉbec City, QuÉbec, Canada, AAAI Press, 2014, pp. 1293–1299.

[16] Yann LeCun, Yoshua Bengio, Geoffrey E. Hinton, Deep learning, Nat. 521 (7553) (2015) 436–444.

[17] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.

[18] Hui Xia, Shu-shu Shao, Chunqiang Hu, Rui Zhang, Tie Qiu, Fu Xiao, Robust clustering model based on attention mechanism and graph convolutional network, IEEE Trans. Knowl. Data Eng. 35 (5) (2023) 5203–5215.

[19] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, Chengqi Zhang, Adversarially regularized graph autoencoder for graph embedding, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, 2018.

[20] Han Zhao, Xu Yang, Zhenru Wang, Erkun Yang, Cheng Deng, Graph debiased contrastive learning with joint representation clustering, in: Zhi-Hua Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021, ijcai.org, 2021, pp. 3434–3440.

[21] Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey E. Hinton, A simple framework for contrastive learning of visual representations, in: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, in: Proceedings of Machine Learning Research, vol. 119, PMLR, 2020, pp. 1597–1607.

[22] Feiping Nie, Ziheng Li, Rong Wang, Xuelong Li, An effective and efficient algorithm for K-means clustering with new formulation, IEEE Trans. Knowl. Data Eng. 35 (4) (2023) 3433–3443.

[23] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, Jin Young Choi, Symmetric graph convolutional autoencoder for unsupervised graph representation learning, in: 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, IEEE, 2019, pp. 6518–6527.

[24] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Chengqi Zhang, Attributed graph clustering: A deep attentional embedding approach, in: Sarit Kraus (Ed.), Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, ijcai.org, 2019, pp. 3670–3676.

[25] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, Peng Cui, Structural deep clustering network, in: Yennun Huang, Irwin King, Tie-Yan Liu, Maarten van Steen (Eds.), WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020, ACM / IW3C2, 2020, pp. 1400–1410.

[26] Wenxuan Tu, Sihang Zhou, Xinwang Liu, Xifeng Guo, Zhiping Cai, En Zhu, Jieren Cheng, Deep fusion clustering network, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, the Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, AAAI Press, 2021, pp. 9978–9987.

[27] Yue Liu, Wenxuan Tu, Sihang Zhou, Xinwang Liu, Linxuan Song, Xihong Yang, En Zhu, Deep graph clustering via dual correlation reduction, in: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, the Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022, AAAI Press, 2022, pp. 7603–7611.

[28] Thomas N. Kipf, Max Welling, Variational graph auto-encoders CoRR, abs/1611.07308, 2016.

[29] Pengwei Hu, Keith C.C. Chan, Tiantian He, Deep graph clustering in social network, in: Rick Barrett, Rick Cummings, Eugene Agichtein, Evgeniy Gabrilovich (Eds.), Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, April 3-7, 2017, ACM, 2017, pp. 1425–1426.

[30] Shaosheng Cao, Wei Lu, Qiongkai Xu, Deep neural networks for learning graph representations, in: Dale Schuurmans, Michael P. Wellman (Eds.), Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA, AAAI Press, 2016, pp. 1145–1152.

[31] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, Jing Jiang, MGAE: Marginalized graph autoencoder for graph clustering, in: Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, Chenliang Li (Eds.), Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017, ACM, 2017, pp. 889–898.

[32] Jinyu Cai, Yi Han, Wenzhong Guo, Jicong Fan, Deep graph-level clustering using pseudo-label-guided mutual information maximization network CoRR, abs/2302.02369, 2023, arXiv:2302.02369.

[33] Barakeel Fanseu Kamhoua, Lin Zhang, Kaili Ma, James Cheng, Bo Li, Bo Han, GRACE: A general graph convolution framework for attributed graph clustering, ACM Trans. Knowl. Discov. Data 17 (3) (2023) 31:1–31:31.

[34] Shuiqiao Yang, Sunny Verma, Borui Cai, Jiaojiao Jiang, Kun Yu, Fang Chen, Shui Yu, Variational co-embedding learning for attributed network clustering, Knowl.-Based Syst. 270 (2023) 110530.

[35] Ruina Bai, Ruizhang Huang, Yongbin Qin, Yanping Chen, Yong Xu, A structural consensus representation learning framework for multi-view clustering, Knowl.-Based Syst. 283 (2024) 111132.
[36] Zhiwen Cao, Xijiong Xie, Feixiang Sun, Jiabei Qian, Consensus cluster structure guided multi-view unsupervised feature selection, Knowl.-Based Syst. 271 (2023) 110578.
[37] Renjie Lin, Shide Du, Shiping Wang, Wenzhong Guo, Multi-view clustering via optimal transport algorithm, Knowl.-Based Syst. 279 (2023) 110954.
[38] Xin Zou, Chang Tang, Xiao Zheng, Kun Sun, Wei Zhang, Deqiong Ding, Inclusivity induced adaptive graph learning for multi-view clustering, Knowl.-Based Syst. 267 (2023) 110424.
[39] Jing Liu, Fuyuan Cao, Xuechun Jing, Jiye Liang, Deep multi-view graph clustering network with weighting mechanism and collaborative training, Expert Syst. Appl. 236 (2024) 121298.
[40] Hui-Jia Li, Yuhao Feng, Chengyi Xia, Jie Cao, Overlapping graph clustering in attributed networks via generalized cluster potential game, ACM Trans. Knowl. Discov. Data 18 (1) (2024) 27:1–27:26.
[41] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. Ser. B Stat. Methodol. 39 (1) (1977) 1–22.
[42] Dilin Wang, Qiang Liu, Nonlinear stein variational gradient descent for learning diversified mixture models, in: Kamalika Chaudhuri, Ruslan Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, in: Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 6576–6585.
[43] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, Stephan Günnemann, Pitfalls of graph neural network evaluation, CoRR abs/1811.05868, 2018.
[44] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, Edward Y. Chang, Network representation learning with rich text information, in: Qiang Yang, Michael J. Wooldridge (Eds.), Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, AAAI Press, 2015, pp. 2111–2117.
[45] Zhihao Peng, Hui Liu, Yuheng Jia, Junhui Hou, Attention-driven graph clustering network, in: Heng Tao Shen, Yueting Zhuang, John R. Smith, Yang Yang, Pablo César, Florian Metze, Balakrishnan Prabhakaran (Eds.), MM '21: ACM Multimedia Conference, Virtual Event, China, October 20 - 24, 2021, ACM, 2021, pp. 935–943.
[46] Xiaxia He, Boyue Wang, Yongli Hu, Junbin Gao, Yanfeng Sun, Baocai Yin, Parallelly adaptive graph convolutional clustering model, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[47] Zhihao Peng, Hui Liu, Yuheng Jia, Junhui Hou, Deep attention-guided graph clustering with dual self-supervision, IEEE Trans. Circuits Syst. Video Technol. 33 (7) (2023) 3296–3307.
[48] László Lovász, Michael D. Plummer, Matching Theory, vol. 367, American Mathematical Soc., 2009.
[49] Laurens Van der Maaten, Geoffrey Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (11) (2008).
[50] T. Konstantin Rusch, Michael M. Bronstein, Siddhartha Mishra, A survey on oversmoothing in graph neural networks, CoRR abs/2303.10993, 2023.
[51] T. Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, Michael M. Bronstein, Graph-coupled oscillator networks, in: Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, Sivan Sabato (Eds.), International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA, in: Proceedings of Machine Learning Research, vol. 162, PMLR, 2022, pp. 18888–18909.
[52] Lingxiao Zhao, Leman Akoglu, PairNorm: Tackling oversmoothing in GNNs, in: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, OpenReview.net, 2020.
[53] Chen Cai, Yusu Wang, A note on over-smoothing for graph neural networks, CoRR abs/2006.13318, 2020.
[54] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, Xu Sun, Measuring and relieving the over-smoothing problem for graph neural networks from the topological view, in: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, the Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, the Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 3438–3445.
[55] Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, Xia Hu, Towards deeper graph neural networks with differentiable group normalization, in: Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, Hsuan-Tien Lin (Eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual, 2020.