1-2024

Deep Adaptive Graph Clustering via Von Mises-Fisher Distributions

Pengfei Wang
Chinese Academy of Sciences

Daqing Wu
Peking University

Chong Chen
DAMO Academy, Alibaba Group

Kunpeng Liu
Portland State University

Yanjie Fu
University of Central Florida

See next page for additional authors

Citation Details

Wang, P., Wu, D., Chen, C., Liu, K., Fu, Y., Huang, J., Zhou, Y., Zhan, J., & Hua, X. (2024). Deep Adaptive Graph Clustering via von Mises-Fisher Distributions. ACM Transactions on the Web, 18(2), 1–21.

Deep Adaptive Graph Clustering via von Mises-Fisher Distributions

PENGFEI WANG, Computer Network Information Center, Chinese Academy of Sciences, China
DAQING WU, Peking University; DAMO Academy, Alibaba Group, China
CHONG CHEN, DAMO Academy, Alibaba Group, China
KUNPENG LIU, Portland State University, USA
YANJIE FU, University of Central Florida, USA
JIANQIANG HUANG, DAMO Academy, Alibaba Group, China
YUANCHUN ZHOU, Computer Network Information Center, Chinese Academy of Sciences, China
JIANFENG ZHAN, Institute of Computing Technology, Chinese Academy of Sciences, China
XIANSHENG HUA, DAMO Academy, Alibaba Group, China

Graph clustering has been a hot research topic and is widely used in many fields, such as community detection in social networks. Lots of works combining auto-encoder and graph neural networks have been applied to clustering tasks by utilizing node attributes and graph structure. These works usually assumed the inherent parameters (i.e., size and variance) of different clusters in the latent embedding space are homogeneous, and hence the assigned probability is monotonous over the Euclidean distance between node embeddings and centroids. Unfortunately, this assumption usually does not hold since the size and concentration of different clusters can be quite different, which limits the clustering accuracy. In addition, the node embeddings in deep graph clustering methods are usually L2 normalized so that it lies on the surface of a unit hyper-sphere. To solve this problem, we proposed **D**eep **A**daptive **G**raph **C**lustering via von Mises-Fisher distributions, namely DAGC. DAGC assumes the node embeddings $\boldsymbol{H}$ can be drawn from a von Mises-Fisher distribution and each cluster k is associated with cluster inherent parameters $\boldsymbol{\rho}_k$ which includes cluster center $\boldsymbol{\mu}$ and cluster cohesion degree κ. Then we adopt an EM-like approach (i.e., $\mathcal{P}(\boldsymbol{H}|\boldsymbol{\rho})$ and $\mathcal{P}(\boldsymbol{\rho}|\boldsymbol{H})$, respectively) to learn the embedding and cluster inherent parameters alternately. Specifically, with the node embeddings, we proposed to update the cluster centers in an attraction-repulsion manner to make the cluster centers more separable. And given the cluster inherent parameters, a likelihood-based loss is proposed to make node embeddings more

P. Wang and D. Wu authors contributed equally to this research.

This research was supported by the Natural Science Foundation of China under Grant No. 61836013, the Strategic Priority Research Program of CAS XDB31000000, and the Chinese Academy of Sciences Network Security and Informatization Special Application Demonstration Project CAS-WX2021SF-0101-03.

Authors' addresses: P. Wang and Y. Zhou, Computer Network Information Center, Chinese Academy of Sciences, CAS Informatization Plaza No. 2 Dong Sheng Nan Lu, Haidian District, Beijing 100083, China; emails: wpf2106@gmail.com, zyc@cnic.cn; D. Wu, Peking University, DAMO Academy, Alibaba Group, 969 West Wen Yi Road, Yu Hang District, Hangzhou 311121, China; email: wudq@pku.edu.cn; C. Chen, J. Huang, and X. Hua, DAMO Academy, Alibaba Group, 969 West Wen Yi Road, Yu Hang District, Hangzhou 311121, China; emails: cheung1990@126.com, {jianqiang.jqh, huaxiansheng}@gmail.com; K. Liu, Portland State University, 1825 SW Broadway, Portland, OR 97201, USA; email: kunpeng@pdx.edu; Y. Fu (corresponding author), University of Central Florida, 4000 Central Florida Blvd, Orlando, FL 32816, USA; J. Zhan, Institute of Computing Technology, Chinese Academy of Sciences, No.6 Kexueyuan South Road Zhongguancun,Haidian District Beijing, China; email: zhanjianfeng@ict.ac.cn.

1559-1131/2024/01-ART22 $15.00
https://doi.org/10.1145/3580521

concentrated around cluster centers. Thus, DAGC can simultaneously improve the intra-cluster compactness and inter-cluster heterogeneity. Finally, extensive experiments conducted on four benchmark datasets have demonstrated that the proposed DAGC consistently outperforms the state-of-the-art methods, especially on imbalanced datasets.

CCS Concepts: • **Mathematics of computing** → *Graph algorithms*; • **Theory of computation** → **Unsupervised learning and clustering**; • **Computing methodologies** → *Learning latent representations;*

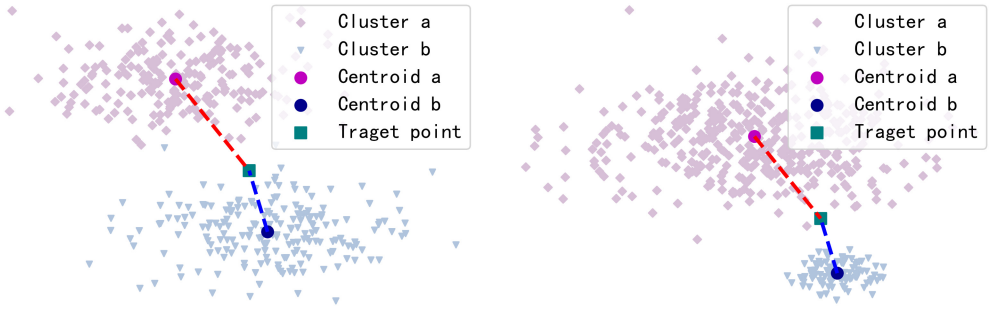
Additional Key Words and Phrases: Graph embedding, graph clustering, vMF

ACM Reference format:
Pengfei Wang, Daqing Wu, Chong Chen, Kunpeng Liu, Yanjie Fu, Jianqiang Huang, Yuanchun Zhou, Jianfeng Zhan, and Xiansheng Hua. 2024. Deep Adaptive Graph Clustering via von Mises-Fisher Distributions. *ACM Trans. Web* 18, 2, Article 22 (January 2024), 21 pages.
https://doi.org/10.1145/3580521

1 INTRODUCTION

The goal of graph clustering is to divide the nodes in a large graph into different clusters such that the inter-cluster similarity is low and the intra-cluster similarity is high [40]. Graph clustering techniques are very useful for detecting connected relationships with nodes' similar properties in a large graph [66], where it is critical to identify the specific patterns or structures efficiently. And deep learning has been widely applied to many tasks and achieved lots of improvements [35], including the clustering approach [45]. In this paper, we focus on the deep graph clustering approach. Recently, deep graph clustering has attracted intensive attention and achieved great success in many fields [48, 49], such as co-saliency [19], community detection [44], and image segmentation [7, 9]. With the emerging representation learning [18, 27, 29, 38, 51], many deep clustering approaches have been proposed to investigate the deep graph clustering efficiency. Specifically, some researchers proposed a method that can learn feature representations and cluster assignments using deep neural networks by partitioning the nodes into several disjoint datasets in [55]. Some researchers jointly consider the local structure preservation in deep clustering, optimizing cluster labels assignment and learning features by integrating the clustering loss and auto-encoder reconstruction loss [14]. The authors in [59] applied the **Gaussian mixture model (GMM)** as the prior in VAE to improve the learned embeddings. The authors in [5] designed a delivery operator and a dual self-supervised mechanism to combine the auto-encoder representation and the **graph convolutional networks (GCNs)** representation. And the authors in [46] proposed a dynamic cross-modality fusion mechanism and a triplet self-supervised strategy, and so on.

However, these existing works are all based on the assumption that the clusters are homogeneous and learned by the Euclidean distance between points in a given feature space, inevitably limiting the representation learning and clustering efficiency. First, most existing deep graph clustering works combined with auto-encoder and graph neural networks do not consider the discrepancy in cluster size. But in many real-world applications from both academia and industry, clusters are usually unequal in size [2, 17, 50, 52], e.g., the communities are imbalanced in social network [1]. Second, each cluster has a particular cluster distribution. That is, given a cluster, the maximum distance between all the points and the centroid is the unique property of the cluster. And the radii of different clusters may be different. But most methods mentioned above do not consider the cluster cohesion degree. For example, in Figure 1, the length of *Red dotted line* represents the Euclidean distance between the *target point* and *centroid a*, denoted as d_a. Likewise, the length of *Blue dotted line*, d_b is the distance between the *target point* and *centroid b*. Obviously, distance d_a is longer than the distance d_b in both Figure 1(a) and Figure 1(b). Then the *target point* will be



(a) Similar cluster inherent parameters (b) Different cluster inherent parameters

Fig. 1. A toy example of clustering. The *target point* will be assigned to the *cluster b* due to the distances between *target point* and the *centroids*. Specifically, it is correct to assign *target point* into *cluster a* when *cluster a* and *cluster b* have similar inherent parameters as shown in Figure 1(a). But the *target point* should be assigned into *cluster a* by considering the different inherent parameters of *cluster a* and *cluster b* as shown in Figure 1(b).

assigned into the *cluster b* as the *target point* is closer to *centroid b*. However, it is not very suitable to assign *target point* into the *cluster b* in the scenario shown in Figure 1(b) without considering the cluster cohesion degree. More specifically, Figure 1(a) shows it is appropriate to assign *target point* into the *cluster b* when *cluster a* and *cluster b* have similar inherent parameters. But on the contrary, as 1(b) shows, without utilizing the inherent parameters of clusters, the nodes may be assigned into inappropriate groups. In addition, the *target point* should be assigned to the *cluster a* by considering the cluster cohesion degree.

Meanwhile, the general idea of the clustering algorithm is to find the best centroid for each cluster. A centroid is the geometric center of a convex object, which can be considered as the generalization of the mean. Given a specific feature space, the clustering method assigns the points into the clusters with the shortest distances between candidate points and centroids. Probabilistic clustering algorithms such as K-means clustering method [16], Multinomial Mixtures [60], and Gaussian Distributions [59], have been used to discover the latent structures and relations in deep graph clustering. However, these assumptions are questionable to be directly used in deep graph clustering to learn the node embeddings. For example, in a citation graph, the documents usually are represented as a point on a unit-sphere. Similarly, in Image-modeling, the unit normalized spatial pyramid vector is a common representation [13]. Thus, the popular clustering assumptions such as Gaussian or Multinomial are not appropriate. Can we develop models that are suitable to model the inherent parameter of clusters on a unit-sphere?

To tackle the challenges mentioned above, inspired by [13], we propose a **deep adaptive graph clustering method (DAGC)** via **von Mises-Fisher (vMF)** distributions. Firstly, in a specific graph, each node represented as a point on a unit-sphere is assumed to be drawn from one vMF distribution. Given a specific cluster, the nodes are drawn from the same vMF distribution. The centroid of the cluster could be modeled by the mean direction of vMF distribution. And the cluster cohesion degree could be modeled by the concentration parameter of vMF distribution. Then, we can model the distribution of the latent embedding space by considering the inherent parameters of clusters. It is almost not possible to obtain the node embeddings and the inherent parameters at the same time. Therefore, we use an EM-like [56] approach to optimize the model. On the E-step, with the node embeddings, we estimate the assigned probability by the posterior of mixture distribution and reassign it by Sinkhorn's theorem [42] to capture the imbalance of cluster size. And then we can adjust the cluster inherent parameters automatically. After that, on the M-step, we

propose to update the node embeddings based on the cluster assignment and the cluster inherent parameters. Given the cluster inherent parameters, we take advantage of a likelihood-based loss function to make the representations in the same cluster more compact. Along this line, DAGC can simultaneously improve the intra-cluster compactness and inter-cluster heterogeneity. We also present detailed experimental comparisons of the proposed algorithms DAGC with the start-of-art methods related to deep graph clustering. Our key contributions can be summarized as follows:

- We propose a deep adaptive graph clustering method via vMF distributions, which can effectively capture the heterogeneity of clusters by modeling the node embeddings with the cluster inherent parameters.
- We design an efficient learning strategy, an EM-like approach, which updates the clustering parameters and node embeddings alternately, which can increase the inter-class heterogeneity and intra-class compactness, respectively.
- We conduct experiments on four challenging real-world graph datasets, the experimental results show our approach can outperform the state-of-the-art deep graph clustering models, especially on imbalanced datasets. Comprehensive ablation experiments have also proved that every component of our method is indispensable.

2 RELATED WORK

2.1 Attributed Graph Embedding

Graph embedding has attracted increasing attention in many applications [4, 11, 24, 30, 31, 57, 61]. Graph embedding, also known as network embedding [6] or network representation learning [62], aims to learn low-dimensional representations for nodes in graphs. In addition, attributed graph embedding methods assume node attribute information is available and exploit both topological information and attribute features simultaneously [12]. TADW [58] proved that DeepWalk can be interpreted as a factorization approach and proposed an extension to DeepWalk to explore node features. DANE [25] deals with the dynamic environment with an incremental matrix factorization approach, and LANE [20] incorporates the label information into the optimization process to learn a better embedding. [43] proposes an attributed graph embedding model with the node/edge attributed information by constructing a heterogeneous graph. [65] proposes a framework to learn node representations from a sequence of temporal interactions with two coupled memory networks to store and update node embeddings in external matrices. [8] design a non-parametric Laplacian smoothing filter that preserves optimal denoising properties to filter out high-frequency noises to learn node embeddings. The authors in [64] integrate both structure and feature information into the kernel matrix via a higher-order graph convolution to make the spectral loss well-adapt to attributed graphs. In [32], the authors treat the protein-protein interaction prediction problem as a link prediction problem in attribute networks, then they use an attributed embedding approach to predict the interactions between proteins in the PPI network. The work in [39] proposes an unsupervised graph embedding method to efficiently capture structural properties as well as node labels and attributes in a graph. Although these algorithms are well designed for graph data, they have largely ignored the node embedding distribution, which may result in poor representation in the real graph data.

2.2 Deep Graph Clustering

Recently, due to the strong representation power of deep neural networks, many deep clustering methods have been proposed and achieved impressive performance [14, 15, 26, 34, 37, 55, 63]. Auto-encoder [18] is one of the most commonly used unsupervised deep neural networks, which plays a crucial role in deep clustering. DEC [55] is the most popular method which used the auto-encoder to learn the deep representations by mining divergence between assignment

distribution and target distribution. To exploit the structural information underlying the data, some GCNs based clustering methods were proposed [5, 23, 33, 36, 46, 48]. [23] proposed using the GAE and VGAE to learn the graph-structured data via iteratively aggregating neighborhood representations around each central node. [48] provided DAEGC to encode the topological structure and node contents by introducing the attentional neighbor-wise fusion strategy on the GAE framework. ARGA adversarially regularized GAE further improved the clustering performance by introducing an adversarial learning scheme to learn the graph embedding [33]. SDCN [5] designed a delivery operator and a dual self-supervised mechanism. [36] proposed an attention-based deep graph clustering method by considering the dynamic fusion strategy and the multi-scale features fusion. DFCN [46] designed a dynamic cross-modality fusion mechanism and a triplet self-supervised strategy. Although these methods improve the clustering performance, they merely concern the design of the backbone but ignore the heterogeneity of clusters in the clustering stage.

3 PRELIMINARIES

In this section, we first present some preliminary graph notations about the graph data. Then we formulate the specific problem setting of graph clustering. Finally, we simply introduce the framework of our proposed approach.

3.1 Notations

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X}, \boldsymbol{A})$, $\mathcal{V} = \{v_1, v_2, v_3, \ldots, v_N\}$ is a set of N nodes in the graph and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set of graph. Each edge e_{ij} in $\mathbb{E}$ describes the connection between two different nodes v_i and v_j, and hence e_{ij} can be represented as (v_i, v_j), where $v_i, v_j \in \mathcal{V}$ and nodes v_i and v_j are adjacent nodes. $\boldsymbol{A} \in \{0, 1\}^{N \times N}$ is the adjacency matrix of a graph, and each element in the adjacency matrix $\boldsymbol{A}$ represents whether or not two nodes are connected in a graph. We denote by an $N \times N$ matrix A for the adjacency matrix of a graph $\mathcal{G}$. Namely, for $\forall v_i, v_j \in \mathcal{V}$, $A_{ij} = 1$ if there exists an edge between node v_i and node v_j, otherwise, $A_{ij} = 0$. We assume there are self-loops in the graph, thus $A_{ii} = 1$ for all i. In addition, $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots, \boldsymbol{x}_N]^T$ is the attribute features of nodes where $\boldsymbol{x}_i \in \mathbb{R}^F$ and F is the total number of node attributes. Given a vector $\boldsymbol{x}$, we write $\|\boldsymbol{x}\|_2$ as the its Euclidean norm. Given a subset $S \subseteq V$, we write $|S|$ as the number of nodes in S.

3.2 Problem Statement

In line with the aforementioned graph notations, given a specific graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X}, \boldsymbol{A})$, the graph clustering methods focus on mapping each node $v_i \in \mathcal{V}$ to the low-dimensional embedding $\boldsymbol{h}_i \in \mathbb{R}^d$ based on its original attributes $\boldsymbol{x}_i \in \mathbb{R}^F$ and the graph structure, and separates the node set $\mathcal{V}$ into K disjoint subsets $\mathcal{V} = \mathcal{V}_1 \bigcup \mathcal{V}_2 \cdots \bigcup \mathcal{V}_K$ such that each $\mathcal{V}_k$ is corresponding to a specific semantic. The main goal of graph embedding is to encode nodes into low-dimensional space while preserving the information of graph structure and node attributes, thus the node similarity in the latent embedded space can approximate the node similarity in the original high-dimensional graph. Then we assume $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3, \ldots, \boldsymbol{h}_N]^T$ is the latent embedding which can preserve the graph structure properties and node pairwise similarity in the embedded latent space facilitates an approximation of the corresponding node similarity in the original space. And we denote the set $\boldsymbol{c} = [c_1, c_2, c_3 \ldots, c_N]$ as the clustering assignment for all nodes, each element c_i can indicate the label for node v_i. We will simultaneously learn the embedding $\boldsymbol{H}$ and clustering assignments $\boldsymbol{c}$ in this paper.

3.3 Framework Overview

To begin with, we assume that each node is drawn from one of the K vMF distributions and each node belongs to a specific cluster in the graph $\mathcal{G}$. Based on this assumption, we propose a deep

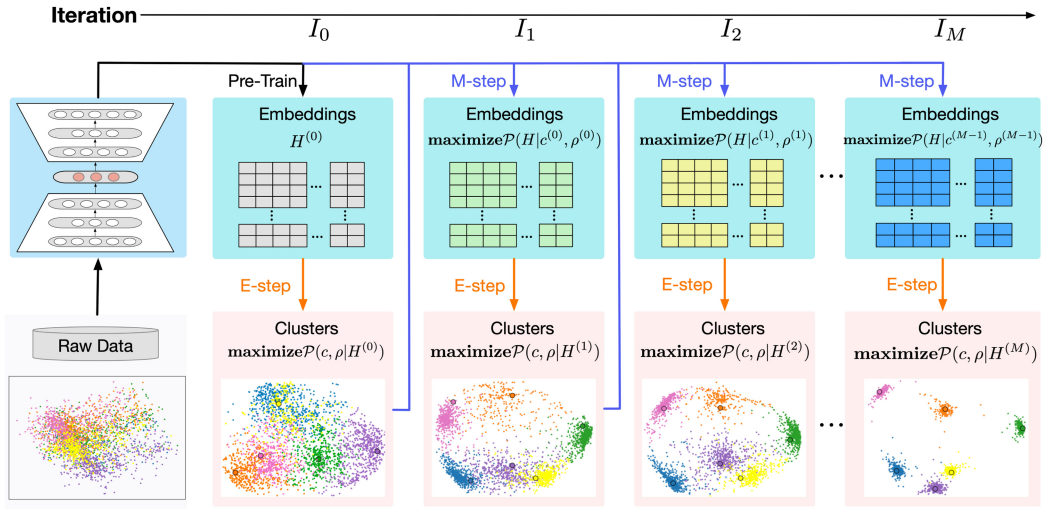


Fig. 2. Framework Overview of Deep Adaptive Graph Clustering via von Mises-Fisher Distributions. The representations of graph attention auto-encoder and parameters of the clusters based on vMF distributions can be updated with an efficient EM-like approach.

adaptive graph clustering model, the framework is illustrated in Figure 2. Specifically, we develop a graph attention auto-encoder as a backbone that can effectively integrate both the graph structure information and node attribute information to learn the hidden representations for all nodes. Then we use an adaptive model to fit the hidden representations via vMF distributions. During iteration, we adopt an efficient EM-like updating approach, which alternatively updates the representations of graph attention auto-encoder and parameters of the clusters based on vMF distributions. More details are given in the following section.

4 METHOD

According to the mathematical problem setting of graph clustering in Section 3, we will introduce the components of the deep adaptive graph clustering approach via Von Mises-Fisher. First, we present the graph attention auto-encoder and give the reconstruction loss of attributes and structure. Then we propose the deep adaptive graph clustering model based vMF distribution. Finally, we show the details of the parameter updating process with an efficient EM-like approach.

4.1 Graph Attention Auto-encoder

Graph attention auto-encoder includes an encoder that maps nodes from the attribute space to the latent space and a decoder performing an inverse mapping. For the sake of neat notation, we denote $\boldsymbol{z}_i^{(l)}$ as the output representation of node v_i in the l-th feed-forward layer, then the attribute feature vector $\boldsymbol{x}_i = \boldsymbol{z}_i^{(0)}$, $\boldsymbol{h}_i = \boldsymbol{z}_i^{(L)}/\|\boldsymbol{z}_i\|_2$ is the hidden representation of node v_i on the unit hypersphere and $\hat{\boldsymbol{x}}_i = \boldsymbol{z}_i^{(2L)}$ is the reconstructed representation.

To represent both graph structure information and node attribute information in a unified framework, we consider **graph attention network (GATs)** [47] as the encoder, i.e.,

$$\boldsymbol{z}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^{(l)} \boldsymbol{W}^{(l)} \boldsymbol{z}_j^{(l)}\right) \tag{1}$$

where $\mathcal{N}_i$ denotes the neighbors of node v_i, σ is a non-linear function, and $\alpha_{ij}^{(l)}$ is the attention coefficient that indicates the importance of neighbor node v_j to node v_i, which can be computed by:

$$\alpha_{ij}^{(l)} = \frac{\exp\Big(\text{LeakyReLU}(\boldsymbol{a}^{(l)^T}[\boldsymbol{W}^{(l)}\boldsymbol{z}_i^{(l)}||\boldsymbol{W}^{(l)}\boldsymbol{z}_j^{(l)}])\Big)}{\sum_{k\in\mathcal{N}_i}\exp\Big(\text{LeakyReLU}(\boldsymbol{a}^{(l)^T}[\boldsymbol{W}^{(l)}\boldsymbol{z}_i^{(l)}||\boldsymbol{W}^{(l)}\boldsymbol{z}_k^{(l)}])\Big)} \tag{2}$$

where $\boldsymbol{a}^{(l)}$ is the parametric weight vector, $||$ is the symbol of concatenation operation, and $(\cdot)^T$ denotes transpose operation.

4.2 Attributes and Structure Reconstruction

In this work, our decoder is a combination of the inner product layer and the encoder symmetric GATs layers following the same propagation style defined in the Equation (1). Attributes reconstruction loss is the basic paradigm of auto-encoder, which minimizes the difference between the input and output of auto-encoder with the following formula:

$$\mathcal{L}_X = \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2 \tag{3}$$

Different from the previous works [33, 46, 48] about the structure reconstruction loss of graph link structure $\boldsymbol{A}$, we minimize the structure reconstruction loss by measuring the difference between $\hat{\boldsymbol{A}}$ and pairwise similarity matrix $\boldsymbol{S}$:

$$\mathcal{L}_A = \frac{1}{N^2}\sum_{i,j=1}^{N}(\hat{\boldsymbol{A}}_{ij} - \boldsymbol{S}_{ij})^2 \tag{4}$$

where $\boldsymbol{S}_{ij} = \frac{\boldsymbol{x}_i{}^T\boldsymbol{x}_j}{\|\boldsymbol{x}_i\|_2\|\boldsymbol{x}_j\|_2}$ is the cosine similarity of attribute features between node v_i and node v_j, $\hat{\boldsymbol{A}}_{ij} = \boldsymbol{h}_i{}^T\boldsymbol{h}_j$ is the cosine similarity of embeddings between node v_i and node v_j. The final reconstruction loss is a hybrid of the content reconstruction loss and the structure reconstruction loss:

$$\mathcal{L}_r = \mathcal{L}_A + \mathcal{L}_X \tag{5}$$

4.3 Adaptive Model based vMF

In this paper, we assume the nodes are drawn from the K **von Mises-Fisher (vMF)** distributions. The von Mises-Fisher (vMF) distribution defines a probability density over points on a unit-sphere. It is parameterized by two parameters, mean parameters μ and concentration parameter κ. μ defines the mean value in the distribution and κ determines the spread of the probability mass around the mean. Specifically, to efficiently capture the variable inter-cluster dispersion and intra-class compactness, we assume the node embeddings $\boldsymbol{H}$ can be drawn from K vMF distributions adeptly. For each cluster $k \in \{1,\ldots,K\}$, we defined the cluster inherent parameters $\boldsymbol{\rho}_k = (\boldsymbol{\mu}_k, \kappa_k)$, where $\boldsymbol{\mu}_k$ is the centroid and κ_k is the magnitude parameter. Thus, $\boldsymbol{\mu}_k$ defines the mean embedding in the cluster k and κ_k determines the spread of the probability mass around the cluster centroid. Then, if any $\boldsymbol{h}$ belongs to cluster k, the probability density function for node representation $\boldsymbol{h} \in \mathbb{R}^d$ is given by following:

$$f(\boldsymbol{h}|\boldsymbol{\rho_k}) = C_d(\kappa_k)\exp(\kappa_k\boldsymbol{\mu}_k^T\boldsymbol{h}), \tag{6}$$

where $\|\boldsymbol{\mu}_k\| = 1$, $\kappa_k \geq 0$ and $C_d(\kappa_k)$ is the normalizer which is expressed as:

$$C_d(\kappa_k) = \frac{\kappa^{\frac{d}{2}} - 1}{(2\pi)^{\frac{d}{2}}\boldsymbol{I}_{\frac{d}{2}-1}(\kappa)}, \tag{7}$$

where $\boldsymbol{I}$ is the modified Bessel function of the first kind. Then given node embeddings $\boldsymbol{H}$, cluster assignment $\boldsymbol{c}$ and cluster inherent parameters $\boldsymbol{\rho} = (\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_K)$, the likelihood function can be written as:

$$\mathcal{P}(\boldsymbol{H}|\boldsymbol{c}, \boldsymbol{\rho}) = \Pi_{i=1}^{N} C_d(\kappa_{c_i}) \exp(\kappa_{c_i} \boldsymbol{\mu}_{c_i}^T \boldsymbol{h}_i). \tag{8}$$

Since $\boldsymbol{H}$, $\boldsymbol{c}$, and $\boldsymbol{\rho}$ are all unknown, it is impossible to infer all of them at one time. Here we will adopt an EM-like updating approach, i.e., alternatively updating $\boldsymbol{H}$, $\boldsymbol{c}$ and $\boldsymbol{\rho}$ by maximizing $\mathcal{P}(\boldsymbol{H}|\boldsymbol{c}, \boldsymbol{\rho})$ and $\mathcal{P}(\boldsymbol{c}, \boldsymbol{\rho}|\boldsymbol{H})$, respectively. By maximizing $\mathcal{P}(\boldsymbol{c}, \boldsymbol{\rho}|\boldsymbol{H})$, it can increase the inter-cluster separability, while it can improve the intra-cluster compactness by maximizing $\mathcal{P}(\boldsymbol{H}|\boldsymbol{c}, \boldsymbol{\rho})$.

4.4 Cluster Assignment

Denote $t \in \{I_0, I_1, I_2, \ldots, I_M\}$ as the iteration index, when given the t-th updated latent embedding $\boldsymbol{H}^{(t)}$, clustering assignment $\boldsymbol{c}^{(t)}$ and cluster inherent parameters $\boldsymbol{\rho}^{(t)}$, we can calculate $\pi_k^{(t)} = \frac{|\boldsymbol{c}^{(t)}=k|}{N}$, where $\pi_k^{(t)}$ can be viewed as the proportion of samples for cluster k. Intuitively, we can calculate the assignment probability matrix $\boldsymbol{P}^{(t)}$ by:

$$\begin{aligned} P_{ik}^{(t)} &= p(c_i = k|\boldsymbol{h}_i^{(t)}; \boldsymbol{\rho}_k^{(t)}) \\ &= \frac{\pi_k^{(t)} f(\boldsymbol{h}_i^{(t)}|\boldsymbol{\rho}_k^{(t)})}{\sum_{k'=1}^{K} \pi_{k'}^{(t)} f(\boldsymbol{h}_i^{(t)}|\boldsymbol{\rho}_{k'}^{(t)})} \end{aligned} \tag{9}$$

where $P_{ik}^{(t)}$ is the probability of i-th node belongs to k-th cluster. For a typical classification problem with deterministic labels, the learning goal can be summarized as the minimization of the average cross-entropy loss. However, node labels are not accessible in unsupervised clustering. Considering that pseudo labels are relaxed to be the posterior probability matrix $\boldsymbol{P}^{(t)}$, where each row represents the cluster assignment probabilities of one node with the schema defined in the Equation (9). And there exist degenerate solutions by assigning all data points to a single (arbitrary) label. To avoid this extreme case, we add the constraints that the label distribution must be consistent with the mixing proportions. Therefore, the updated posterior probability matrix $\boldsymbol{Q}^{(t)}$ should satisfy the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{Q}^{(t)}} \quad & -\boldsymbol{Q}^{(t)} \log \boldsymbol{P}^{(t)} - \frac{1}{\lambda} \mathrm{H}(\boldsymbol{Q}^{(t)}) \\ \text{s.t.} \quad & \boldsymbol{Q}^{(t)} \in \mathbb{R}_+^{N \times C}, \\ & \boldsymbol{Q}^{(t)} \mathbf{1}_C = \mathbf{1}_N \ \text{ and } \ \boldsymbol{Q}^{(t)^T} \mathbf{1}_N = N\boldsymbol{\pi}^{(t)} \end{aligned} \tag{10}$$

where H is the entropy function and λ is the smoothness parameter that can control the equilibrium of clusters. Apparently, the existence and unicity of the solution are guaranteed by Equation (10). Furthermore, Sinkhorn's theorem [42] states that there exist diagonal matrices $\mathrm{diag}(\boldsymbol{u})$ and $\mathrm{diag}(\boldsymbol{v})$ such that $\mathrm{diag}(\boldsymbol{u})\boldsymbol{P}^{(t)\lambda}\mathrm{diag}(\boldsymbol{v})$ has i-th row sum 1 and c-th column sum $N\pi_c^{(t)}$ and can be computed with Sinkhorn's fixed point iteration. In addition, the posterior probability updating process can be shown in Algorithm 1.

4.5 E-step: Updating c and ρ

Given the latent embedding $\boldsymbol{H}^{(t)}$ and the assignment probability matrix $\boldsymbol{Q}^{(t)}$ of iteration t, we can update $\boldsymbol{c}$ and $\boldsymbol{\rho}$ by

$$\text{maximize } \mathcal{P}(\boldsymbol{c}, \boldsymbol{\rho}|\boldsymbol{H}^{(t)}).$$

ALGORITHM 1: The posterior probability updating algorithm

Require: The posterior probability $\boldsymbol{P}^{(t)}$;
Row sum constraint $\mathbf{1}_N$;
Column sum constraint $N\boldsymbol{\pi}^{(t)}$;
Ensure: The updated posterior probability matrix $\boldsymbol{Q}^{(t)}$.
1: Initialize $\boldsymbol{u} = \mathbf{1}_N$ and $\boldsymbol{v} = \mathbf{1}_C$.
2: **loop**
3: $\quad \boldsymbol{u} = \boldsymbol{I}_N(\boldsymbol{P}^{(t)\lambda}\boldsymbol{v})^{-1}$;
4: $\quad \boldsymbol{v} = \mathrm{diag}(N\boldsymbol{\pi}^{(t)})(\boldsymbol{P}^{(t)\lambda^T}\boldsymbol{u})^{-1}$;
5: **end loop**
6: $\boldsymbol{Q}^{(t)} = \mathrm{diag}(\boldsymbol{u})\boldsymbol{P}^{(t)\lambda}\mathrm{diag}(\boldsymbol{v})$.

Updating cluster assignment $\boldsymbol{c}$. Similar to the EM algorithm for the Gaussian mixture model, we can update $\boldsymbol{c}$ simply by

$$c_i^{(t+1)} = \arg\max_k Q_{ik}^{(t)} \tag{11}$$

Updating cluster center $\boldsymbol{\mu}$. Generally, the sum of distances between data points and the corresponding center is regarded as the objective to measure whether centers are preferable. We aim to find the optimal cluster centers closed to associating data points, i.e.,

$$\begin{aligned} \min_{\{\boldsymbol{\mu}_k\}_1^K} \quad & \sum_{k=1}^{K}\sum_{i=1}^{N} Q_{ik}^{(t)} \|\boldsymbol{h}_i^{(t)} - \boldsymbol{\mu}_k\|_2^2 \\ \text{s.t.} \quad & \|\boldsymbol{\mu}_k\|_2 = 1 \quad k = 1, 2, \ldots, K \end{aligned} \tag{12}$$

If $Q_{ik}^{(t)}$ is binary and follows the hard-assignment scheme, the solution of the above optimization problem is the centroids estimation in spherical k-means [10]. If $Q_{ik}^{(t)}$ follows the soft-assignment scheme, with the gradient descent method, the updated center can be computed by

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\boldsymbol{\mu}_k^{(t)} + \eta \cdot \nabla\boldsymbol{\mu}_k^{(t)}}{\|\boldsymbol{\mu}_k^{(t)} + \eta \cdot \nabla\boldsymbol{\mu}_k^{(t)}\|_2} \tag{13}$$

$$\nabla\boldsymbol{\mu}_k^{(t)} = \sum_{i=1}^{N} Q_{ik}^{(t)}(\boldsymbol{h}_i^{(t)} - \boldsymbol{\mu}_k^{(t)}) \tag{14}$$

where η is the updating rate. Note that when we set updating rate as $\eta = 1/\sum_{i=1}^{N} Q_{ik}^{(t)}$, the updating scheme degenerates to the centroids updating method in [3]. However, the node embeddings keep changing during the learning process, it is unsuitable to update the center so quickly. Note that the updating strategy of Equation (14) would cause different centers which collapse to one data point, which is harmful to the node embedding learning. Besides, from the Equation (14), we can observe that if node v_i has a high posterior probability on the k-th cluster, then it has a strong attraction to pull $\boldsymbol{\mu}_k^{(t)}$ with displacement distance $(\eta \cdot Q_{ik}^{(t)}; \Theta^{(t)})(\boldsymbol{h}_i^{(t)} - \boldsymbol{\mu}_k^{(t)})$. Therefore, for each cluster, we encourage the center to move close to the data points with the high posterior probabilities and away from the data points with the low posterior probabilities, i.e.,

$$\begin{aligned} \nabla\boldsymbol{\mu}_k^{(t)} = & \sum_{Q_{ik}^{(t)} \geq \tau_k^{(t)}} Q_{ik}^{(t)}(\boldsymbol{h}_i^{(t)} - \boldsymbol{\mu}_k^{(t)}) \\ & - \sum_{Q_{ik}^{(t)} < \tau_k^{(t)}} Q_{ik}^{(t)}(\boldsymbol{h}_i^{(t)} - \boldsymbol{\mu}_k^{(t)}) \end{aligned} \tag{15}$$

where $\tau_k^{(t)}$ is the $N\pi_k^{(t)}$-largest probability in the k-th column of probability matrix $\boldsymbol{Q}^{(t)}$.

Updating cluster cohesion degree $\boldsymbol{\kappa}$. As for the concentration parameter, the larger value of $\kappa_k^{(t)}$ implies a higher cohesion degree of the cluster. In particular, when $\kappa_k^{(t)} = 0$, $f(\boldsymbol{h}_i^{(t)}|\boldsymbol{\mu}_k^{(t)}, \kappa_k^{(t)})$ reduces to the uniform density, and as $\kappa_k^{(t)} \to \infty$, $f(\boldsymbol{h}_i^{(t)}|\boldsymbol{\mu}_k^{(t)}, \kappa_k^{(t)})$ degenerated to one-point distribution. Additionally, inspired by the concentration parameter estimation in [3], we utilize the reasonable updating formulation as follows

$$\kappa_k^{(t+1)} = \frac{A_d(\kappa_k^{(t)})d - A_d(\kappa_k^{(t)})^3}{1 - A_d(\kappa_k^{(t)})^2} \tag{16}$$

where $A_d(\kappa_k^{(t)}) = \frac{I_{d/2}(\kappa_k^{(t)})}{I_{d/2-1}(\kappa_k^{(t)})}$.

4.6 M-step: Updating H

Given cluster assignment $\boldsymbol{c}^{(t+1)}$ and cluster inherent parameters $\boldsymbol{\rho}^{(t+1)}$, we will update $\boldsymbol{H}$ by

$$\text{maximize } \mathcal{P}(\boldsymbol{H}|\boldsymbol{c}^{(t+1)}, \boldsymbol{\rho}^{(t+1)}).$$

It is equivalent to minimize the following loss function:

$$\mathcal{L}_p = -\frac{1}{N}\sum_{i=1}^{N} \kappa_{c_i^{(t+1)}} \boldsymbol{\mu}_{c_i^{(t+1)}}^T \boldsymbol{h} \tag{17}$$

By including the construction loss, the overall loss function for updating $\boldsymbol{H}$

$$\mathcal{L} = \mathcal{L}_r + \gamma \mathcal{L}_p = \mathcal{L}_A + \mathcal{L}_X + \gamma \mathcal{L}_p \tag{18}$$

where γ is a hyper-parameter that balances the weight of reconstruction loss and prediction loss. Then we can obtain $\boldsymbol{H}^{(t+1)}$ by SGD algorithm.

4.7 Overall Algorithm

In practice, we first pre-train the graph attention auto-encoder in a reconstruction task, then conduct K-means on the node embeddings to initialize the clustering parameters. After that, to learn more discriminative node representations, we leverage an alternate learning strategy. When fixing the node embeddings, we update the clustering parameters to adjust the deflected mixture distribution. Given the current clustering parameters, we update the parameters of graph attention auto-encoder by minimizing the overall loss $\mathcal{L}$. This process enables the learned node embeddings close to their associating cluster centroids. The details are summarized in the Algorithm 2.

4.8 Complexity Analysis

Along the proposed model DAGC, we denote the dimensions of layers in graph attention auto-encoder as $d_1, d_2, \ldots, d_{2L}$, then the time complexity of graph attention auto-encoder can be expressed as $O(N(d_1F + d_2d_1 + \cdots + d_{2L}d_{2L-1}) + |\mathcal{E}|(d_1 + d_2 + \cdots + d_{2L-1}))$, where F is the total number of node attributes. And the time complexity of the mixture parameter learning process is $O(NKd)$, where N is the total number of nodes, K is the number of divided clusters, and d is the dimension of latent embedding. Since attribute number F, dimensions of auto-encoder $d_1, d_2, \ldots, d_{2L}$, and parameters K, d can be regarded as constants, the overall time complexity is linearly related to the numbers of nodes and edges.

ALGORITHM 2: Deep Adaptive Graph Clustering via vMF Distributions

Require: Attribute feature matrix $\boldsymbol{X}$;
Graph adjacent matrix $\boldsymbol{A}$;
Number of clusters K;
The number of iteration M.
Ensure: Clustering results $\boldsymbol{c} = \{c_i\}_{i=1}^{N}$.

```
Pre-train graph attention auto-encoder by minimizing the final loss as shown in Equation (5).
Conduct K-Means on the node embeddings learned by the pre-trained auto-encoder.
Initialize cluster centers {μ_k}_{k=1}^K with K-Means.
Initialize clustering assignment {c_k}_{k=1}^K by the hard assignment and initialize the proportion
of clusters {π_k}_{k=1}^K.
Initialize concentration parameter {κ_k}_{k=1}^K by the average sample-based parameters.
for iteration from 1 to M do
    Generate node embedding H^(t);
    Compute the posterior probability matrix P^(t);
    Compute the updated posterior probability matrix Q^(t) through Sinkhorn's fixed point iter-
    ation;
    update c^(t+1) by the Equation (11);
    update μ^(t+1) by the Equation (13);
    update κ^(t+1) by the Equation (16);
    update the parameters of graph attention auto-encoder by minimizing Equation (18)
end for
Return the clustering results c^(M).
```

5 EXPERIMENT

In order to show the effectiveness of our proposed model. In this section, we first introduce the four public benchmark datasets widely used in graph clustering tasks. Then we show the compared baselines and the evaluation metrics used in this paper. In addition, we present the implementation details and show the performance of the proposed model. Finally, we also conducted some additional experiments to show the effectiveness of our proposed model including an ablation study, visualization, parameter sensitivity, and efficiency analysis.

5.1 Datasets

Our proposed DAGC is evaluated on four public benchmark datasets including multiple types of graphs. The statistical information of these datasets is provided in Table 1 and the detailed descriptions are the followings:

- **ACM**[1] [53]: This is a paper network from the ACM dataset. There is an edge between two papers if they are written by the same author. Paper features are the bag-of-words of the keywords. We select papers published in KDD, SIGMOD, SIGCOMM, and MobiCOMM and divide the papers into three classes (database, wireless communication, data mining) by their research area.
- **DBLP**[2] [53]: This is an author network from the DBLP dataset. There is an edge between the two authors if they are the co-author relationship. The authors are divided into four areas:

[1] https://dl.acm.org/.
[2] https://dblp.uni-trier.de.

Table 1. The Statistics of the Datasets

Dataset	Network Type	Nodes	Classes	Dimension
ACM	Paper	3025	3	1870
DBLP	Author	4058	4	334
Citeseer	Citation	3327	6	3703
Amazon	Item	7650	8	745

database, data mining, machine learning, and information retrieval. We label each author's research area according to the conferences they submitted. Author features are the elements of a bag-of-words represented by keywords.

- **Citeseer**[3] [22]: This is a citation network that contains sparse bag-of-words feature vectors for each document, and a list of citation links between documents. The labels contain six areas: agents, artificial intelligence, database, information retrieval, machine language, and HCI.
- **Amazon**[4] [41]: This is an item co-purchased network, where nodes represent goods, edges indicate that two goods are frequently bought together, node features are bag-of-words encoded product reviews, and class labels are given by the product category.

5.2 Baselines

We compare the performance of our proposed method with seven baseline methods:

- **K-means** [16]: A classical clustering method based on the raw data.
- **AE** [18]: It performs K-means on the representations learned by auto-encoder.
- **DEC** [55]: It employs a clustering loss to supervise the process of clustering with the auto-encoder backbone.
- **DAEGC**[5] [48]: It uses an attention network to learn the node representations and employs a clustering loss to supervise the process of graph clustering.
- **SDCN**[6] [5]: It is representative of hybrid methods which take advantage of both AE and GCN modules for clustering.
- **AGCN**[7] [36]: It utilizes the attention-based method by considering the dynamic fusion strategy and the multi-scale features fusion.
- **DFCN**[8] [46]: It designs a dynamic cross-modality fusion mechanism and a triplet self-supervised strategy.

5.3 Metrics

To show the effectiveness of the proposed method, we employ four popular metrics [54]. For each metric, the larger value implies a better clustering result. The best map between cluster ID and class ID is found by using the Kuhn-Munkres algorithm [28]. The four specific evaluation metrics are as follows:

- **ACC: Accuracy** shows the quality between the predicted labels and the true labels. After achieving the best map between the class ID and the cluster ID by using the Kuhn-Munkres

[3] https://citeseerx.ist.psu.edu/index.
[4] https://www.amazon.com/.
[5] https://github.com/kouyongqi/DAEGC.
[6] https://github.com/bdy9527/SDCN.
[7] https://github.com/ZhihaoPENG-CityU/AGCN.
[8] https://github.com/WxTu/DFCN.

Table 2. Clustering Results on Four Benchmark Datasets (mean±std)

Dataset	Metric	K-Means	AE	DEC	DAEGC	SDCN	AGCN	DFCN	DAGC
ACM	ACC	$67.31_{\pm 0.71}$	$81.83_{\pm 0.08}$	$84.33_{\pm 0.76}$	$86.94_{\pm 2.83}$	$90.45_{\pm 0.18}$	$90.59_{\pm 0.15}$	$90.84_{\pm 0.15}$	$\mathbf{92.02_{\pm 0.12}}$
	NMI	$32.44_{\pm 0.46}$	$49.30_{\pm 0.16}$	$54.54_{\pm 1.51}$	$56.18_{\pm 4.15}$	$68.31_{\pm 0.25}$	$68.38_{\pm 0.45}$	$69.39_{\pm 0.36}$	$\mathbf{71.68_{\pm 0.11}}$
	ARI	$30.60_{\pm 0.69}$	$54.64_{\pm 0.16}$	$60.64_{\pm 1.87}$	$59.35_{\pm 3.89}$	$73.91_{\pm 0.40}$	$74.20_{\pm 0.38}$	$74.93_{\pm 0.37}$	$\mathbf{77.77_{\pm 0.14}}$
	F1	$67.57_{\pm 0.74}$	$82.01_{\pm 0.08}$	$84.51_{\pm 0.74}$	$87.07_{\pm 2.79}$	$90.42_{\pm 0.19}$	$90.58_{\pm 0.17}$	$90.78_{\pm 0.16}$	$\mathbf{92.04_{\pm 0.12}}$
DBLP	ACC	$38.65_{\pm 0.65}$	$51.43_{\pm 0.35}$	$58.16_{\pm 0.56}$	$62.05_{\pm 0.48}$	$68.05_{\pm 1.81}$	$73.26_{\pm 0.37}$	$76.02_{\pm 0.77}$	$\mathbf{81.46_{\pm 0.19}}$
	NMI	$11.45_{\pm 0.38}$	$25.40_{\pm 0.16}$	$29.51_{\pm 0.28}$	$32.49_{\pm 0.45}$	$39.50_{\pm 1.34}$	$39.68_{\pm 0.42}$	$43.65_{\pm 1.01}$	$\mathbf{52.51_{\pm 0.41}}$
	ARI	$6.97_{\pm 0.39}$	$12.21_{\pm 0.43}$	$23.92_{\pm 0.39}$	$21.03_{\pm 0.52}$	$39.15_{\pm 2.01}$	$42.49_{\pm 0.31}$	$46.95_{\pm 1.51}$	$\mathbf{58.28_{\pm 0.36}}$
	F1	$31.92_{\pm 0.27}$	$52.53_{\pm 0.36}$	$59.38_{\pm 0.51}$	$61.75_{\pm 0.67}$	$67.71_{\pm 1.51}$	$72.80_{\pm 0.56}$	$75.74_{\pm 0.75}$	$\mathbf{80.10_{\pm 0.21}}$
Citeseer	ACC	$39.32_{\pm 3.17}$	$57.08_{\pm 0.13}$	$55.89_{\pm 0.20}$	$64.54_{\pm 1.39}$	$65.96_{\pm 0.31}$	$68.79_{\pm 0.23}$	$69.54_{\pm 0.15}$	$\mathbf{70.60_{\pm 0.06}}$
	NMI	$16.94_{\pm 3.22}$	$27.64_{\pm 0.08}$	$28.34_{\pm 0.30}$	$36.41_{\pm 0.86}$	$38.71_{\pm 0.32}$	$41.54_{\pm 0.30}$	$43.93_{\pm 0.22}$	$\mathbf{44.85_{\pm 0.19}}$
	ARI	$13.43_{\pm 3.02}$	$29.31_{\pm 0.14}$	$28.12_{\pm 0.36}$	$37.78_{\pm 1.24}$	$40.17_{\pm 0.43}$	$43.79_{\pm 0.31}$	$45.45_{\pm 0.26}$	$\mathbf{47.05_{\pm 0.18}}$
	F1	$36.08_{\pm 3.53}$	$53.80_{\pm 0.11}$	$52.62_{\pm 0.17}$	$62.20_{\pm 1.32}$	$63.62_{\pm 0.24}$	$62.37_{\pm 0.21}$	$64.27_{\pm 0.20}$	$\mathbf{65.87_{\pm 0.06}}$
Amazon	ACC	$43.24_{\pm 4.37}$	$59.72_{\pm 3.87}$	$59.84_{\pm 0.24}$	$71.56_{\pm 3.34}$	$75.51_{\pm 1.92}$	$76.80_{\pm 0.40}$	$79.13_{\pm 0.90}$	$\mathbf{84.95_{\pm 0.08}}$
	NMI	$30.74_{\pm 4.33}$	$51.89_{\pm 3.70}$	$54.67_{\pm 0.30}$	$60.68_{\pm 2.58}$	$63.26_{\pm 2.05}$	$63.17_{\pm 0.72}$	$71.12_{\pm 0.98}$	$\mathbf{74.05_{\pm 0.17}}$
	ARI	$17.78_{\pm 2.82}$	$40.47_{\pm 3.06}$	$42.21_{\pm 0.25}$	$52.05_{\pm 3.76}$	$54.95_{\pm 2.23}$	$55.67_{\pm 0.84}$	$62.41_{\pm 1.58}$	$\mathbf{69.43_{\pm 0.20}}$
	F1	$30.34_{\pm 7.45}$	$47.76_{\pm 6.04}$	$47.72_{\pm 2.87}$	$67.55_{\pm 3.39}$	$69.44_{\pm 1.34}$	$68.32_{\pm 0.62}$	$72.92_{\pm 0.81}$	$\mathbf{83.17_{\pm 0.09}}$

algorithm, clustering accuracy can be computed by $ACC = \frac{\sum_{n=1}^{N} \mathcal{I}_n}{N}$, where $\mathcal{I}_n$ is an indicator function, $\mathcal{I}_n = 1$ when the predicted label and the true label are the same, and $\mathcal{I}_n = 0$ otherwise.

- **NMI: Normalized Mutual Information**, a symmetric index computing the similarity between two clustering solutions based on the confusion matrix (also referred to as the contingency matrix).
- **ARI: Adjusted Rand Index**, ARI shows the ratio of the number of node pairs similarly classified in both solutions, divided by the total number of pairs. It compares two clusterings with the number of cluster membership agreements and disarrangements.
- **F1: F1** score can combine the precision and recall into a single metric by taking their harmonic mean with equation $F1 = \frac{2*Precision*Recall}{Precision+Recall}$, where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$. Similar to ACC, the macro F1-score can be computed after achieving the best map between the class ID and the cluster ID with the Kuhn-Munkres algorithm.

5.4 Implementation Details

In the experiments, we implement our proposed model based on PyTorch. For baseline methods, we report the results listed in their papers. The embedding size d is fixed to 16 for all datasets, which is suitable for the model to learn strong representations [5, 48]. We optimize DAGC with Adam [21] optimizer having a learning rate 0.005 for all datasets, weight decay 0.005 for ACM and Citeseer, $5e^{-4}$ for DBLP and Amazon. And the number of epochs is fixed to 100. In terms of updating rate of centroid η, we set it as $1e^{-3}$ for ACM, $5e^{-4}$ for DBLP and Citeseer, $1e^{-5}$ for Amazon. As two introduced hyper-parameters loss balance coefficient and smoothness parameter, we apply grid search, and $\{\gamma, \lambda\}$ are set to $\{0.5, 10\}$ for ACM, $\{1, 3\}$ for DBLP, $\{0.7, 5\}$ for Citeseer and $\{1, 5\}$ for Amazon.

5.5 Overall Clustering Performance

The clustering results of DAGC are averaged over 10 runs with random seeds, and we report the mean values and the corresponding standard deviations. The overall results are shown in Table 2. We have the following observations:

- For each metric, our method DAGC achieves the best results in all four datasets. In particular, compared with the best results of the baselines, our approach achieves a significant

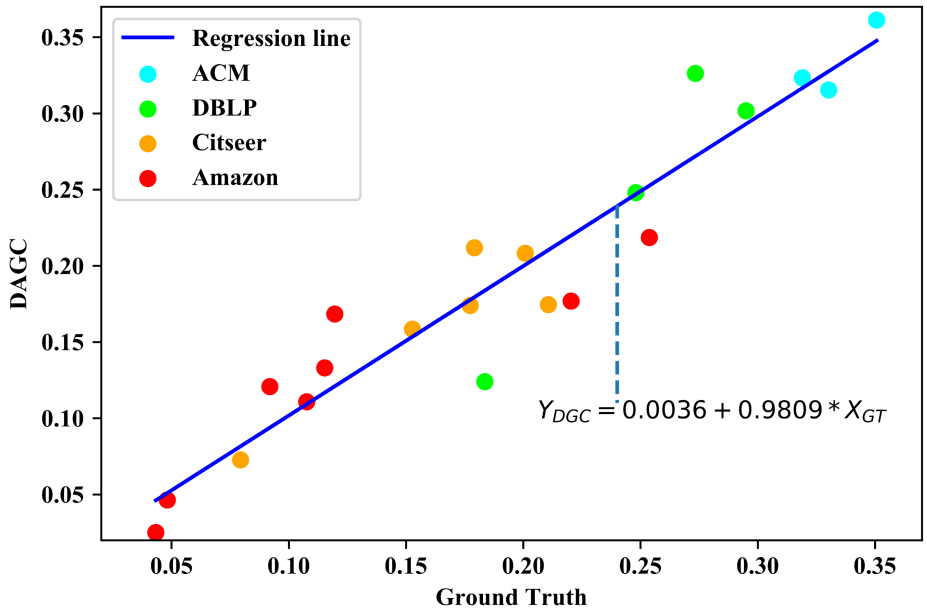


Fig. 3. Correlation demonstration of cluster proportion distribution between ground truth and DAGC.

improvement of 4.35% on ACC, 7.45% on NMI, 10.65% on ARI, and 5.95% on F1 score, averagely. Different from other methods such as DFCN and AGCN which focus on fusing the attributes and graph structure, DAGC pays more attention to the clustering process. It uses a deep adaptive model to handle the latent embedding and adopts an EM-like updating approach, which can simultaneously improve the intra-cluster compactness and inter-cluster heterogeneity. This is why DAGC can achieve better performance than state-of-the-art baselines.

- For two imbalanced datasets DBLP and Amazon, DAGC obtains a remarkable improvement of 9.03% on ACC, 12.21% on NMI, 17.69% on ARI, and 9.89% on F1, averagely. Existing deep graph clustering methods are incompetent in dealing with imbalanced datasets and intrinsically tend to produce balanced clusters. On the contrary, DAGC takes cluster inherent parameters (both cluster size and intra-cluster variance) into consideration and can automatically estimate the latent parameters. As shown in Figure 3, the estimated cluster size by DAGC is highly consistent with the ground-truth, the performance shows our model can capture the inter-cluster dispersion and intra-compactness even on imbalanced datasets, it also demonstrates the superiority of DAGC.

5.6 Ablation Study

We conduct ablation studies to evaluate the contributions of the centroid updating strategy, mixing proportion, and concentration parameter. Particularly, we introduce the following model variants: DAGC-h takes hard-assignment scheme and updates centroids with the means of data points; DAGC-s utilizes a soft-assignment strategy and replaces the binary patterns with the posterior probability; and DAGC-t considers the Student's t-distribution given in DEC to compute the posterior probability. DAGC-π sets mixing proportion $\pi_c = 1/C$; DAGC-κ regards cohesion coefficient κ_c as the inverse of the variance of all data points.

Table 3. Ablation Study on Centroid Updating Strategy

Dataset	Metric	DAGC-h	DAGC-s	DAGC
ACM	ACC	$90.44_{\pm 0.12}$	$90.98_{\pm 0.11}$	$\mathbf{92.02}_{\pm \mathbf{0.12}}$
	NMI	$68.32_{\pm 0.19}$	$68.84_{\pm 0.16}$	$\mathbf{71.68}_{\pm \mathbf{0.11}}$
	ARI	$74.87_{\pm 0.14}$	$75.01_{\pm 0.13}$	$\mathbf{77.77}_{\pm \mathbf{0.12}}$
	F1	$90.94_{\pm 0.09}$	$91.02_{\pm 0.12}$	$\mathbf{92.04}_{\pm \mathbf{0.12}}$
DBLP	ACC	$79.64_{\pm 0.16}$	$80.56_{\pm 0.24}$	$\mathbf{81.46}_{\pm \mathbf{0.19}}$
	NMI	$50.47_{\pm 0.34}$	$50.70_{\pm 0.28}$	$\mathbf{52.51}_{\pm \mathbf{0.41}}$
	ARI	$54.79_{\pm 0.25}$	$55.96_{\pm 0.47}$	$\mathbf{58.28}_{\pm \mathbf{0.21}}$
	F1	$78.07_{\pm 0.49}$	$79.69_{\pm 0.27}$	$\mathbf{80.10}_{\pm \mathbf{0.21}}$
Citeseer	ACC	$68.69_{\pm 0.09}$	$69.36_{\pm 0.05}$	$\mathbf{70.06}_{\pm \mathbf{0.06}}$
	NMI	$42.66_{\pm 0.16}$	$43.48_{\pm 0.18}$	$\mathbf{44.85}_{\pm \mathbf{0.19}}$
	ARI	$44.41_{\pm 0.13}$	$45.52_{\pm 0.19}$	$\mathbf{47.05}_{\pm \mathbf{0.18}}$
	F1	$64.51_{\pm 0.07}$	$64.57_{\pm 0.04}$	$\mathbf{65.87}_{\pm \mathbf{0.06}}$
Amazon	ACC	$80.74_{\pm 0.14}$	$82.77_{\pm 0.13}$	$\mathbf{84.95}_{\pm \mathbf{0.08}}$
	NMI	$70.28_{\pm 0.24}$	$72.32_{\pm 0.16}$	$\mathbf{74.05}_{\pm \mathbf{0.17}}$
	ARI	$62.44_{\pm 0.23}$	$65.43_{\pm 0.11}$	$\mathbf{69.43}_{\pm \mathbf{0.20}}$
	F1	$79.80_{\pm 0.18}$	$81.80_{\pm 0.10}$	$\mathbf{83.17}_{\pm \mathbf{0.09}}$

Table 4. Ablation Study on Mixing Proportion and Cohesion Degree

Dataset	Metric	DAGC-t	DAGC-π	DAGC-κ	DAGC
ACM	ACC	$90.94_{\pm 0.15}$	$91.09_{\pm 0.05}$	$91.44_{\pm 0.18}$	$\mathbf{92.02}_{\pm \mathbf{0.12}}$
	NMI	$68.65_{\pm 0.47}$	$69.49_{\pm 0.29}$	$70.30_{\pm 0.58}$	$\mathbf{71.68}_{\pm \mathbf{0.11}}$
	ARI	$74.93_{\pm 0.38}$	$75.33_{\pm 0.14}$	$76.28_{\pm 0.47}$	$\mathbf{77.77}_{\pm \mathbf{0.12}}$
	F1	$90.98_{\pm 0.15}$	$91.12_{\pm 0.05}$	$91.48_{\pm 0.17}$	$\mathbf{92.04}_{\pm \mathbf{0.12}}$
DBLP	ACC	$79.67_{\pm 0.21}$	$81.13_{\pm 0.16}$	$80.73_{\pm 0.33}$	$\mathbf{81.46}_{\pm \mathbf{0.19}}$
	NMI	$50.20_{\pm 0.23}$	$51.82_{\pm 0.19}$	$51.07_{\pm 0.50}$	$\mathbf{52.51}_{\pm \mathbf{0.41}}$
	ARI	$53.85_{\pm 0.31}$	$57.03_{\pm 0.32}$	$56.56_{\pm 0.65}$	$\mathbf{58.28}_{\pm \mathbf{0.21}}$
	F1	$79.29_{\pm 0.24}$	$79.65_{\pm 0.15}$	$79.71_{\pm 0.38}$	$\mathbf{80.10}_{\pm \mathbf{0.21}}$
Citeseer	ACC	$66.60_{\pm 0.07}$	$67.22_{\pm 0.08}$	$69.12_{\pm 0.09}$	$\mathbf{70.06}_{\pm \mathbf{0.06}}$
	NMI	$41.33_{\pm 0.08}$	$41.83_{\pm 0.19}$	$43.30_{\pm 0.09}$	$\mathbf{44.85}_{\pm \mathbf{0.19}}$
	ARI	$42.04_{\pm 0.11}$	$42.84_{\pm 0.15}$	$45.76_{\pm 0.12}$	$\mathbf{47.05}_{\pm \mathbf{0.18}}$
	F1	$63.96_{\pm 0.08}$	$64.28_{\pm 0.08}$	$65.28_{\pm 0.11}$	$\mathbf{65.87}_{\pm \mathbf{0.06}}$
Amazon	ACC	$78.61_{\pm 0.39}$	$80.31_{\pm 0.35}$	$82.85_{\pm 0.15}$	$\mathbf{84.95}_{\pm \mathbf{0.08}}$
	NMI	$68.32_{\pm 0.30}$	$69.98_{\pm 0.12}$	$72.15_{\pm 0.10}$	$\mathbf{74.05}_{\pm \mathbf{0.17}}$
	ARI	$58.23_{\pm 0.35}$	$61.12_{\pm 0.89}$	$65.20_{\pm 0.17}$	$\mathbf{69.43}_{\pm \mathbf{0.20}}$
	F1	$77.45_{\pm 0.48}$	$79.24_{\pm 0.55}$	$81.94_{\pm 0.11}$	$\mathbf{83.17}_{\pm \mathbf{0.09}}$

Table 4 and 3 show the following observations :

- First, compared to DAGC-h, DAGC-s improves the performances due to the fact that the soft-assignment strategy considers the otherness with the probability if two data points are assigned to the same cluster;
- Second, DAGC has acceptable improvement on DAGC-s, which indicates that our centroid updating strategy can estimate the preferable centroids via pushing cluster centroids away from low confident points.

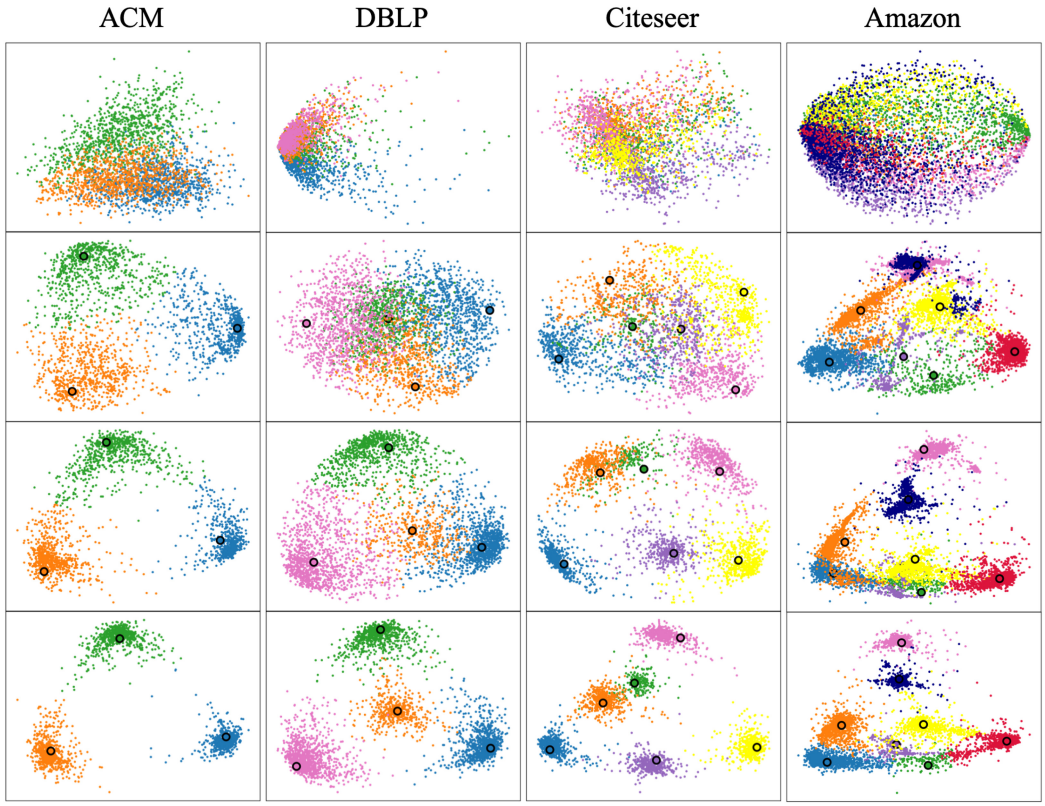


Fig. 4. 2D visualization on four benchmark datasets. Black circles indicate the cluster centroids.

- Finally, Table 4 shows, that both DAGC-π and DAGC-κ jointly considering cluster size and cluster cohesion outperform DAGC-t based on the simple distance measure, but they are not as good as DAGC in comparison.

Therefore, it is essential to consider the cluster inherent parameters for learning the node embeddings.

5.7 Visualization

In order to show the superiority of the representation obtained by our proposed method, PCA is utilized to visualize the feature space. The visualizations on four datasets are given in Figure 4. From up to down, they are the space of raw data, initialization embeddings, and learned embeddings (epoch 10 and epoch 100) of DAGC, respectively. We can see that the representations obtained by DAGC are discriminative, and each cluster is compact. The discriminate cluster distributions indicate the clusters could be distinguished clearly in the feature space.

5.8 Parameter Sensitivity

In order to demonstrate the robustness of the proposed model, we further study the influence of hyper-parameters including loss balance coefficient γ and smoothness parameter λ. Figure 5 and Figure 6 illustrate the effect of γ and λ varying from 0.1 to 10 and 1 to 50, respectively. Specifically, for loss balance coefficient γ, our method performs stably over a wide range of its values as shown

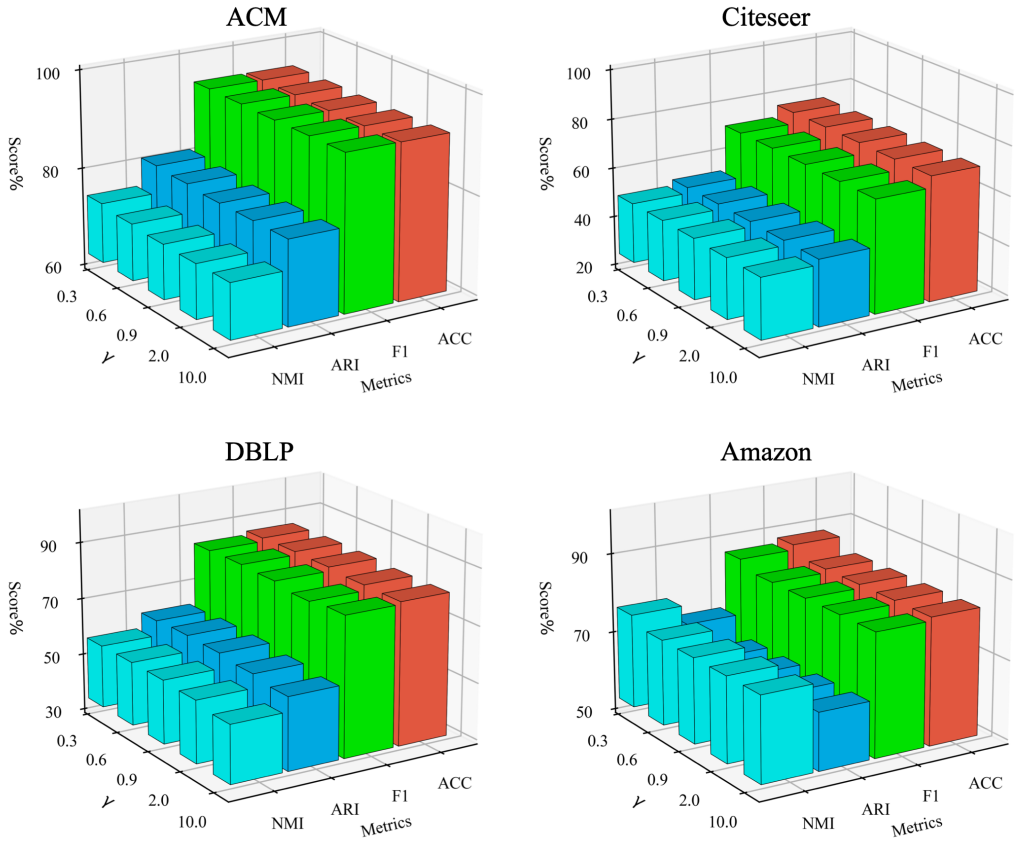


Fig. 5. Performance of DAGC on four benchmark datasets w.r.t. different loss balance coefficient γ.

Table 5. Running Time of the Proposed Model DAGC and the Baseline Model DFCN (mean±std, Time Unit: Second)

Dataset	ACM	DBLP	Citeseer	Amazon
DFCN	$18.50_{\pm 0.22}$	$29.51_{\pm 0.27}$	$37.52_{\pm 0.27}$	$68.54_{\pm 0.56}$
DAGC	$\mathbf{18.42_{\pm 0.21}}$	$\mathbf{25.78_{\pm 0.38}}$	$\mathbf{26.87_{\pm 0.31}}$	$\mathbf{65.67_{\pm 0.45}}$

in Figure 5. And since λ is the smoothness parameter controlling the equilibrium of clusters and the dataset Amazon is more imbalanced than other datasets. Thus, the parameter λ is more sensitive on the dataset Amazon, and there is a small peak in the Amazon data set as shown in Figure 6. In other cases, the performance is relatively stable.

5.9 Efficiency Analysis

To show the efficiency of the proposed method, we compare the running times of the proposed model DAGC and one representative baseline model DFCN [46]. Specifically, the unit of running time we used in this paper is seconds. Table 5 shows that the running times of the models have the same order of magnitude. In addition, the proposed model DAGC requires less running time compared with DFCN.

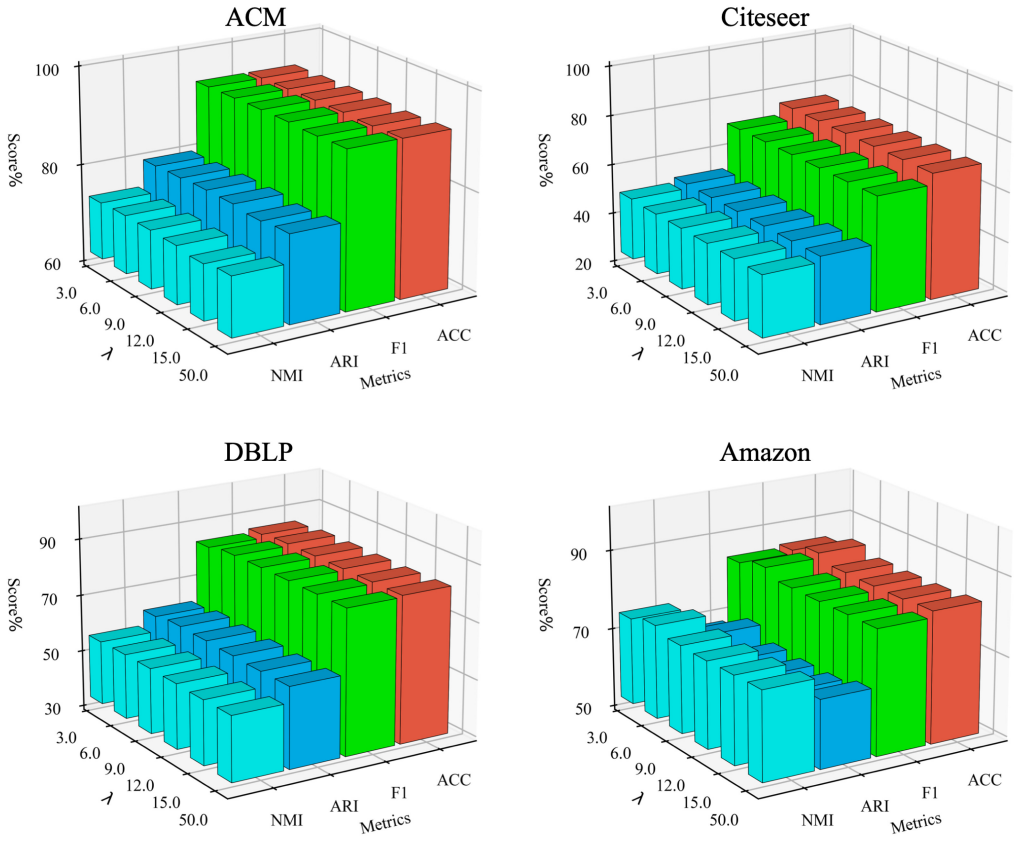


Fig. 6. Performance of DAGC on four benchmark datasets w.r.t. different smoothness parameter λ.

6 CONCLUSION

In this paper, we study the deep graph clustering problem. To address the issue that the density of different clusters can be quite different, we proposed a new method named deep adaptive graph clustering via vMF distributions. Specifically, we model the cluster distribution via considering the cluster inherent properties to better evaluate the distances between candidates and centroids by assuming the nodes can be drawn from vMF distributions. Then, we design an EM-like clustering parameters updating strategy to adjust the mixture distribution guiding the embedding learning. Finally, extensive experiments on four benchmark datasets have been conducted to demonstrate the proposed DAGC consistently outperforms the state-of-the-art methods, especially on imbalanced datasets. In the future, we will develop our method on large-scale or multi-relational graph datasets for the clustering task.

REFERENCES

[1] Cem Aksoylar, Jing Qian, and Venkatesh Saligrama. 2016. Clustering and community detection with imbalanced clusters. *IEEE Transactions on Signal and Information Processing over Networks* 3 (2016), 61–76.

[2] Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. 2017. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*. 1961–1971.

[3] Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. 2005. Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research* 6 (2005), 1345–1382.

[4] Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. 2021. Graph convolution for semi-supervised classification: Improved linear separability and out-of-distribution generalization. *arXiv preprint arXiv:2102.06966* (2021).
[5] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. 2020. Structural deep clustering network. In *Proceedings of the World Wide Web Conference (WWW)*. 1400–1410.
[6] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30 (2018), 1616–1637.
[7] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 132–149.
[8] Ganqu Cui, Jie Zhou, Cheng Yang, and Zhiyuan Liu. 2020. Adaptive graph encoder for attributed graph embedding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 976–985.
[9] Bert De Brabandere, Davy Neven, and Luc Van Gool. 2017. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551* (2017).
[10] Inderjit S. Dhillon and Dharmendra S. Modha. 2001. Concept decompositions for large sparse text data using clustering. *Machine Learning* 42 (2001), 143–175.
[11] Yun Fu and Yunqian Ma. 2012. *Graph Embedding for Pattern Analysis*. Springer Science & Business Media.
[12] Hongchang Gao and Heng Huang. 2018. Deep attributed network embedding. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 3364–3370.
[13] Siddharth Gopal and Yiming Yang. 2014. Von Mises-Fisher clustering models. In *International Conference on Machine Learning*. PMLR, 154–162.
[14] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. 2017. Improved deep embedded clustering with local structure preservation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 1753–1759.
[15] Kai Han, Andrea Vedaldi, and Andrew Zisserman. 2019. Learning to discover novel visual categories via deep transfer clustering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 8401–8409.
[16] John A. Hartigan and Manchek A. Wong. 1979. A k-means clustering algorithm. *Applied Stats* 28 (1979), 100–108.
[17] Haibo He and Edwardo A. Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering* 21 (2009), 1263–1284.
[18] Geoffrey Hinton and Ruslan R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313 (2006), 504–507.
[19] Rongyao Hu, Zhenyun Deng, and Xiaofeng Zhu. 2021. Multi-scale graph fusion for co-saliency detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 7789–7796.
[20] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*. 731–739.
[21] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[22] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[23] Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. *ArXiv abs/1611.07308* (2016).
[24] Chen Li, Xutan Peng, Yuhang Niu, Shanghang Zhang, Hao Peng, Chuan Zhou, and Jianxin Li. 2021. Learning graph attention-aware knowledge graph embedding. *Neurocomputing* 461 (2021), 516–529.
[25] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. 387–396.
[26] Peizhao Li, Han Zhao, and Hongfu Liu. 2020. Deep fair clustering for visual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 9070–9079.
[27] K. Liu, Y. Fu, P. Wang, L. Wu, and X. Li. 2019. Automating feature subspace exploration via multi-agent reinforcement learning. In *the 25th ACM SIGKDD International Conference*.
[28] L. Lovász and M. D. Plummer. 1986. *Matching Theory*. (1986).
[29] Gongxu Luo, Jianxin Li, Jianlin Su, Hao Peng, Carl Yang, Lichao Sun, Philip S. Yu, and Lifang He. 2021. Graph entropy guided node embedding dimension selection for graph neural networks. *arXiv preprint arXiv:2105.03178* (2021).
[30] Muhammad Muzzamil Luqman, Jean-Yves Ramel, Josep Lladós, and Thierry Brouard. 2013. Fuzzy multilevel graph embedding. *Pattern Recognition* 46, 2 (2013), 551–565.
[31] Mario Manzo. 2019. KGEARSRG: Kernel graph embedding on attributed relational sift-based regions graph. *Machine Learning and Knowledge Extraction* 1, 3 (2019), 55.
[32] Elahe Nasiri, Kamal Berahmand, Mehrdad Rostami, and Mohammad Dabiri. 2021. A novel link prediction algorithm for protein-protein interaction networks by attributed graph embedding. *Computers in Biology and Medicine* 137 (2021), 104772.

[33] Shirui Pan, Ruiqi Hu, Sai-Fu Fung, Guodong Long, Jing Jiang, and Chengqi Zhang. 2020. Learning graph embedding with adversarial training methods. *IEEE Transactions on Cybernetics* 50, 6 (2020), 2475–2487.
[34] Hao Peng, Ruitong Zhang, Yingtong Dou, Renyu Yang, Jingyi Zhang, and Philip S. Yu. 2021. Reinforced neighborhood selection guided multi-relational graph neural networks. *ACM Transactions on Information Systems (TOIS)* 40, 4 (2021), 1–46.
[35] Hao Peng, Ruitong Zhang, Shaoning Li, Yuwei Cao, Shirui Pan, and Philip Yu. 2022. Reinforced, incremental and cross-lingual event detection from social messages. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).
[36] Zhihao Peng, Hui Liu, Yuheng Jia, and Junhui Hou. 2021. Attention-driven graph clustering network. *arXiv preprint arXiv:2108.05499* (2021).
[37] Z. Qiao, Y. Du, Y. Fu, P. Wang, and Y. Zhou. 2019. Unsupervised author disambiguation using heterogeneous graph convolutional network embedding. In *2019 IEEE International Conference on Big Data (Big Data)*.
[38] Z. Qiao, P. Wang, Y. Fu, Y. Du, and Y. Zhou. 2020. Tree Structure-Aware Graph Representation Learning via Integrated Hierarchical Aggregation and Relational Metric Learning.
[39] Saurabh Sawlani, Lingxiao Zhao, and Leman Akoglu. 2021. Fast attributed graph embedding via density of states. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 559–568.
[40] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer Science Review* 1, 1 (2007), 27–64.
[41] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).
[42] Richard Sinkhorn. 1967. Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly* 74 (1967), 402–405.
[43] Guolei Sun and Xiangliang Zhang. 2019. A novel framework for node/edge attributed graph embedding. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 169–182.
[44] Heli Sun, Fang He, Jianbin Huang, Yizhou Sun, Yang Li, Chenyu Wang, Liang He, Zhongbin Sun, and Xiaolin Jia. 2020. Network embedding for community detection in attributed networks. *ACM Transactions on Knowledge Discovery from Data* 14 (2020), 1–25.
[45] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. 2014. Learning deep representations for graph clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 28.
[46] Wenxuan Tu, Sihang Zhou, Xinwang Liu, Xifeng Guo, Zhiping Cai, En Zhu, and Jieren Cheng. 2021. Deep fusion clustering network. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 9978–9987.
[47] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
[48] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Attributed graph clustering: A deep attentional embedding approach. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 3670–3676.
[49] Jinghua Wang and Jianmin Jiang. 2021. Unsupervised deep clustering via adaptive GMM modeling and optimization. *Neurocomputing* 433 (2021), 199–211.
[50] P. Wang, Y. Fu, G. Liu, W. Hu, and C. Aggarwal. 2017. Human mobility synchronization and trip purpose detection with mixture of Hawkes processes. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.
[51] P. Wang, Y. Fu, J. Zhang, P. Wang, Y. Zheng, and C. C. Aggarwal. 2018. You are how you drive: Peer and temporal-aware representation learning for driving behavior analysis. *ACM* (2018).
[52] P. Wang, G. Liu, Y. Fu, Y. Zhou, and J. Li. 2018. Spotting trip purposes from taxi trajectories: A general probabilistic model. *ACM Transactions on Intelligent Systems* 9, 3 (2018), 29.1–29.26.
[53] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous graph attention network. In *Proceedings of the World Wide Web Conference (WWW)*. 2022–2032.
[54] Rongkai Xia, Yan Pan, Lei Du, and Jian Yin. 2014. Robust multi-view spectral clustering via low-rank and sparse decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2149–2155.
[55] Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *Proceedings of the International Conference on Machine Learning (ICML)*. 478–487.
[56] Zhipeng Xie, Rui Dong, Zhengheng Deng, Zhenying He, and Weidong Yang. 2013. A probabilistic approach to latent cluster analysis. In *Twenty-third International Joint Conference on Artificial Intelligence*.
[57] Mengjia Xu. 2021. Understanding graph embedding methods and their applications. *SIAM Rev.* 63, 4 (2021), 825–853.
[58] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. 2015. Network representation learning with rich text information. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2111–2117.
[59] Linxiao Yang, Ngai-Man Cheung, Jiaying Li, and Jun Fang. 2019. Deep clustering by Gaussian mixture variational autoencoders with graph embedding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6440–6449.

[60] Jianhua Yin and Jianyong Wang. 2014. A dirichlet multinomial mixture model-based approach for short text clustering. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 233–242.
[61] Zelin Zang, Siyuan Li, Di Wu, Jianzhu Guo, Yongjie Xu, and Stan Z. Li. 2021. Unsupervised deep manifold attributed graph embedding. *arXiv preprint arXiv:2104.13048* (2021).
[62] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. Network representation learning: A survey. *IEEE Transactions on Big Data* 6 (2018), 3–28.
[63] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. 2019. Attributed graph clustering via adaptive graph convolution. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 4327–4333.
[64] Xiaotong Zhang, Han Liu, Xiao-Ming Wu, Xianchao Zhang, and Xinyue Liu. 2021. Spectral embedding network for attributed graph clustering. *Neural Networks* 142 (2021), 388–396.
[65] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhao Li, and Can Wang. 2020. Learning temporal interaction graph embedding via coupled memory networks. In *Proceedings of the Web Conference 2020*. 3049–3055.
[66] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment* 2, 1 (2009), 718–729.

Received 1 February 2022; revised 3 November 2022; accepted 3 December 2022