Contents lists available at ScienceDirect

Expert Systems With Applications

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

Contrastive graph clustering with adaptive filter

Xuanting Xie [a], Wenyu Chen [a,*], Zhao Kang [a,*], Chong Peng [b]

[a] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, China
[b] College of Computer Science and Technology, Qingdao University, Qingdao, 266071, China

ARTICLE INFO

Keywords:
Node clustering
Graph learning
Contrastive learning
High-pass filter

ABSTRACT

Graph clustering has received significant attention in recent years due to the breakthrough of graph neural networks (GNNs). However, GNNs frequently assume strong data homophily, which is not true in many real-world applications. Furthermore, practical graphs are typically noisy and sparse, which inevitably degrades the clustering performance. To this end, we propose a novel Contrastive Graph Clustering (CGC) method with adaptive filter framework. We first design an adaptive filter that can automatically learn a suitable filter for different data, mining holistic information beyond low-frequency components and encoding topology structure information into features. Afterward, we learn a refined graph based on a graph-level contrastive mechanism, which further boosts graph discriminability. Extensive experiments show that the proposed CGC method achieves significant improvement over state-of-the-art methods on several benchmark datasets. In particular, our simple method, which does not employ neural networks, outperforms many deep learning approaches.

不可避免地降低簇性能

整体的

1. Introduction

A graph or network is a data structure for characterizing complex interdependencies in real-world systems. For example, biological networks consist of nodes that make up the elements of a biological system and interconnected edges between elements; social networks' nodes refer to individuals or organizations and connections reflect various social relationships. An attributed graph is more powerful because its nodes are also described by features (Liu, Wen, Kang, Luo, & Tian, 2021). Graph clustering, which finds communities or groups in a graph, is a key technique in many real-world applications. Most conventional clustering methods, such as K-means (Huang, Kang, Xu, & Liu, 2021) and hierarchical clustering (Murtagh & Contreras, 2012), are limited to Euclidean data. However, an attributed graph is typical non-Euclidean data and most classical clustering algorithms are not applicable.

Graph clustering, which is a fully unsupervised problem, has attracted significant attention in recent years and many methods have been proposed. Most graph neural network (GNN)-based methods adopt an embedding approach that seeks a low-dimensional representation of nodes by incorporating the structure information. Afterward, these embeddings are fed into downstream tasks. In particular, a graph autoencoder (GAE) (Tian, Gao, Cui, Chen, & Liu, 2014) is a popular backbone for performing graph embedding, which is then proceeded with various classical clustering techniques. Recently, various variants have been developed, such as adversarial regularized GAE (ARGA) and adversarial regularized variational GAE (ARVGA) (Pan et al., 2019), whose embedding follows a prior distribution.

Inspired by the success of GNNs, several GNNs-based clustering methods have been proposed. End-to-end variational graph clustering (EVGC) (Guo & Dai, 2021) and deep neighbor-aware embedded node clustering (DNENC) (Wang et al., 2022) are the most recent progress that produces state-of-the-art results. EVGC focuses on maintaining the local structure of a graph, whereas DNENC aggregates both the graph structure and node information about a node's neighbors. However, GNNs typically assume the case where all neighboring nodes have high similarity, which is untrue in practice and may damage the performance of downstream tasks. Real-world data are complex and diverse, where both weak and strong homophily can exist in different node neighborhoods (Jin et al., 2021; Zhu et al., 2020). For example, different types of amino acids are prone to create connections in protein structures, and fraudsters are more likely to link with accomplices than other fraudsters in online shopping networks (Pandit, Chau, Wang, & Faloutsos, 2007). Current mainstream methods use polynomial approximated filters, for example, GCN (Kipf & Welling, 2017) applies a first-order approximation of ChebNet (Defferrard, Bresson, & Vandergheynst, 2016) and is considered a low-pass filter (Balcilar et al., 2020). GCN and many follow-up studies (Klicpera, Bojchevski, & Günnemann, 2018; Wu et al., 2019) assume that neighborhoods in a graph share many commonalities, limiting the application of GNNs in general data. Some methods have been developed to handle the heterogeneity problem in node classification tasks (Li, Kim and Wang, 2021; Wu, Pan, Long, Jiang, & Zhang, 2022).

异配性

* Corresponding author.
E-mail addresses: 202122080208@std.uestc.edu.cn (X. Xie), cwy@uestc.edu.cn (W. Chen), zkang@uestc.edu.cn (Z. Kang), cpeng@qdu.edu.cn (C. Peng).

https://doi.org/10.1016/j.eswa.2023.119645
Received 10 August 2022; Received in revised form 28 December 2022; Accepted 29 January 2023
Available online 2 February 2023

In addition, one common assumption of the aforementioned methods is that the input graph is optimal, which is typically violated in real-world applications. Therefore, the performance of these methods depends heavily on the quality of the input data. Real-world graphs are typically noisy or incomplete, which will deteriorate the clustering performance (Chen, Wu and Zaki, 2020). As a result, a high-quality graph for these methods is essential. Several methods have been proposed to filter out noise and learn a high-quality graph. For instance, MAGC (Lin, Kang, Zhang, & Tian, 2023), SMC (Liu, Chen et al., 2022), MCGC (Pan & Kang, 2021) and FGC (Kang et al., 2022) apply low-pass filter to node features to achieve smooth embeddings and learn consensus graphs for multi-view graph data. This type of filtering treats all high-frequency information as noise and removes them. Nonetheless, Bo, Wang, Shi, and Shen (2021), Li, Kim et al. (2021) and Shuman, Narang, Frossard, Ortega, and Vandergheynst (2013) show that high-frequency components can hold critical information about local discontinuity, thus, a low-pass filter restrains the generalizability of a model.

To overcome the aforementioned limitation, we propose a novel framework of Contrastive Graph Clustering (CGC) with adaptive filter. CGC contains two major parts: the first part is a mixed filter that combines both low-pass and high-pass filters; the second one is graph learning. In particular, the contrastive learning idea is applied to draw positive samples (similar nodes) close and push negative samples (dissimilar nodes) apart (Hadsell, Chopra, & LeCun, 2006), which further improves the discriminability of node embeddings. Thus, this results in more accurate graph edge learning. After obtaining the learned graph, we apply spectral clustering on it. To sum up, the main contributions of this study are as follows:

- We propose an adaptive filter, which keeps valuable low -frequency and high-frequency information in the data. Graph filtering encodes topological structure information into features, which results in a smooth representation. To our best knowledge, this is the first study to investigate high-frequency components for graph clustering.
- Graph contrastive regularizer is applied to refine the graph structure. The graph learning approach outputs a high-quality graph for subsequent clustering task.
- Experimental results on benchmark datasets indicate that the proposed shallow method outperforms existing graph clustering methods, including recent deep approaches.

2. Related work

2.1. Graph clustering

Graph clustering methods typically comprise two steps: embedding learning and clustering. How to learn a good representation is the key. Traditional network embedding approaches are unsuitable for large-scale network operations because their complexity is at least quadratic to the number of vertices, thus Large-scale Information Network Embedding (LINE) (Tang et al., 2015) is proposed to solve this problem, which preserves information locally and globally. To efficiently incorporate both structure and content information into an attributed graph, an autoencoder has been adopted in many methods. Graph Autoencoder (GAE) and Variational GAE (VGAE) (Kipf & Welling, 2016) use autoencoder to reconstruct the adjacency matrix. Marginalized GAE (MGAE) (Wang, Pan, Long, Zhu and Jiang, 2017) uses autoencoder to reconstruct node features. These approaches can only catch the neighbors of each node within a certain number of hops and thus may miss the global cluster structure of huge networks. By stacking multiple layers of GNN, Structural Deep Clustering Network (SDCN) (Bo et al., 2020) transfers the representations learned by an autoencoder and captures higher-order structural information. However, the number of training parameters in these deep models is often large, which often leads to overfitting. In AGC (Zhang, Liu, Li, & Wu, 2019), unlike multi-layer superposition such as GCN (Kipf & Welling, 2017), a k-order graph convolution is designed to obtain a smooth feature representation by performing low-pass filtering on node features. This method achieves a smoothed representation by aggregating features of k-hop neighbors iteratively to improve downstream task performance (Ma, Kang, Luo, Tian, & Chen, 2020). Filtering in this way is simple and effective.

2.2. Contrastive learning

Contrastive learning is one of the self-supervised learning methods, which learns discriminative characteristics of data by maximizing the similarity of positive pairs and the distance of negative pairs. It exhibits significant performance even compared with supervised learning methods. SimCLR (Chen, Kornblith, Norouzi and Hinton, 2020) constructs self-supervised samples by data augmentation, and regards images from different views in a mini-batch as negative pairs. BYOL (Grill, Strub, Altché, Tallec, Richemond, Buchatskaya, Doersch, Avila Pires, Guo, Gheshlaghi Azar, et al., 2020) is free of negative samples. Several contrastive learning methods on graphs have also been proposed. GraphCL (You et al., 2020) designs four data augmentation types and learns unsupervised representations of graph data. MVGRL (Hassani & Khasahmadi, 2020) compares two diffusion matrices converted from adjacency matrix to achieve representation learning. GCA (Zhu et al., 2021) proposes a joint data augmentation scheme that is adaptive to the graph structure and attributes. However, most of them assume that the given graph is reliable and require data augmentation to find positive pairs. Recent research on contrastive graph learning by maximizing mutual information (MI) between node and graph representations has yielded state-of-the-art results. For example, Deep Graph Infomax (DGI) (Veličković et al., 2019) allows node representations to hold more global information. Contrastive clustering designs a dual contrastive learning paradigm that performs contrastive learning at both instance and cluster levels (Li, Hu et al., 2021). SCAGC (Xia, Gao, Yang, & Gao, 2021) uses a clustering module to directly produce clustering labels by comparing the representation of distinct clusters. DCRN (Liu, Tu et al., 2022) proposes a siamese network to reduce the information correlation dually and performs K-means on embeddings. However, these approaches assume high quality of graph, which is typically violated in practice. Instead, our method learns a new graph from the original data, upon which a traditional clustering method is applied.

2.3. Graph filtering

Several studies have shown that a smooth graph signal contains more low-frequency base components than high-frequency ones (Zhang et al., 2019). Bruna, Zaremba, Szlam, and LeCun (2014) and Henaff, Bruna, and LeCun (2015) were the first to apply localized spectral filters to the graph structure in Fourier space. Recently, a low-pass filter has been extensively used to obtain a smooth representation (Kang et al., 2022; Zhang et al., 2019), where connected nodes will have similar attributes. However, they fail to capture meaningful high-frequency information and simply remove them as noise. Some studies have highlighted that high-frequency components carry useful information (Chang et al., 2021). AutoGCN (Wu et al., 2022) uses three graph convolutional filters to capture more information, and limits the choice of filter functions within linear and quadratic forms of the graph Laplacian matrix. Li, Kim et al. (2021) and Wu et al. (2022) also find the importance of high frequency band of graph signals, and propose flexible GNNs models, which perform well on both homophilic and heterophilic graphs for node classification. However, these approaches are based on neural networks. Unlike them, the proposed CGC method can be directly applied on top of other clustering methods without deep learning.

$(\varepsilon+1)I-L$

$(\varepsilon+1)I+L$

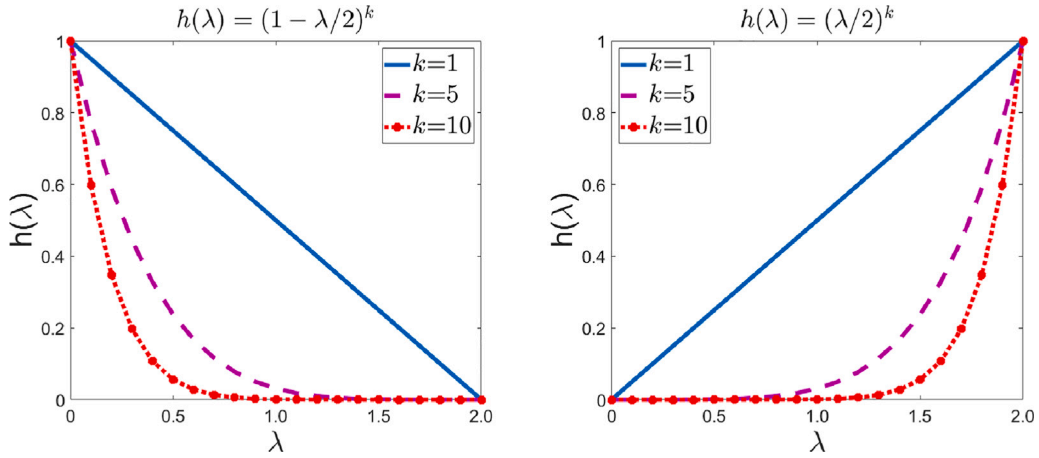

Fig. 1. The filter functions $h(\lambda)=(1-\frac{\lambda}{2})^k$ and $h(\lambda)=(\frac{\lambda}{2})^k$.

3. Methodology

3.1. Notation

Given an undirected graph $\mathcal{G}=(\mathcal{V},E,X)$, where $\mathcal{V}$ represents a set of N nodes, $X=\{x_1,\dots,x_N\}^\top$ represents the feature matrix, and E represents the edge denoted by an adjacency matrix A. If there is an edge between nodes i and j, $a_{ij}=1$; otherwise, $a_{ij}=0$. $D=(A+I)\mathbf{1}_N$ represents the degree matrix, where each node is subjected to a self-loop, and I represents an identity matrix with proper size. The symmetric normalized adjacency matrix is defined as $\widetilde{A}=D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}$ and its graph Laplacian is $L=I-\widetilde{A}$.

合适的大小

3.2. Graph filtering

The feature matrix $X\in\mathbb{R}^{N\times d}$ of N nodes can be considered d N-dimensional graph signals. A signal can be smoothed on neighboring nodes with a spectral filter. Specifically, a series of eigenvalues can be obtained by decomposing the Laplacian matrix of the graph, and a graph convolution filter can be defined as a function of these eigenvalues. $L=U\Lambda U^\top$ denotes the eigendecomposition, $\Lambda=\mathrm{diag}[\lambda_1,\ \lambda_2,\ \dots,\ \lambda_N]$ represents the diagonal matrix of eigenvalues and $\lambda_i\in[0,2]$. A spectral filter on the graph signal x can be expressed as follows:

$$h(L)x=Uh(\Lambda)U^\top x=Udiag[h(\lambda_1),h(\lambda_2),\dots,h(\lambda_N)]U^\top x, \tag{1}$$

合适地

where $h(\lambda)$ represents the filter function. An appropriate $h(\lambda)$ is the core for graph filtering. Let $\bar{X}$ denote the smoothed signal, and $\bar{X}_{i,\cdot}$ represents the ith row of $\bar{X}$. The smoothness of $\bar{X}$ can be measured by Dong, Thanou, Rabbat, and Frossard (2019):

$$\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i}(\bar{X}_{i,\cdot}-\bar{X}_{j,\cdot})^2=2\bar{X}^\top L\bar{X}, \tag{2}$$

where $\mathcal{N}_i$ denotes the neighbors of node i. Thus, we can achieve a smoothed signal $\bar{X}$ by solving the following optimization problem:

$$\min_{\bar{X}}\|\bar{X}-X\|_F^2+\frac{1}{2}Tr(\bar{X}^\top L\bar{X}), \tag{3}$$

求导计算

where the first term is to let $\bar{X}$ be close to X, and the second term attempts to smoothen each node's feature with its neighborhood information. $\bar{X}$ can be calculated by taking the derivative of the objective function and setting it to zero, which results in the equation:

$$\bar{X}=(I+\frac{L}{2})^{-1}X. \tag{4}$$

一阶Taylor级数展开，规避矩阵求逆

We approximate $\bar{X}$ by its first-order Taylor series expansion to avoid matrix inversion, i.e., $\bar{X}=(I-\frac{L}{2})X$. kth order graph filtering can be written as follows:

$$\bar{X}=(I-\frac{L}{2})^kX, \tag{5}$$

where k represents a nonnegative integer. It is easy to see that:

$$(I-\frac{L}{2})^kx=Udiag[(1-\frac{\lambda_1}{2})^k,(1-\frac{\lambda_2}{2})^k,\dots,(1-\frac{\lambda_N}{2})^k]U^\top x. \tag{6}$$

As shown in Fig. 1, $h(\lambda)=(1-\frac{\lambda}{2})^k$ is a low-pass filter, which filters out high-frequency components. Compared with GCN (Kipf & Welling, 2017), which trains convolutional parameters, this filtering technology linearly aggregates neighbor information, making it simpler and easier to use. In real-world applications, the high-frequency part denotes signals that change abruptly and characterize discontinuities (Li, Kim et al., 2021), which is crucial to some tasks. To capture the distinction between a node's information and its neighbors' information (Zhu & Koniusz, 2020), we design another filter function, i.e., $h(\lambda)=(\frac{\lambda}{2})^k$, which is a high-pass filter as shown in Fig. 1. Similarly,

$$\begin{aligned}h(L)x&=Udiag[(\frac{\lambda_1}{2})^k,(\frac{\lambda_2}{2})^k,\dots,(\frac{\lambda_N}{2})^k]U^\top x\\&=\frac{1}{2^k}Udiag[\lambda_1^k,\lambda_2^k,\dots,\lambda_N^k]U^\top x\\&=(\frac{L}{2})^kx.\end{aligned} \tag{7}$$

To capture the entire spectral information about both local smoothness and discontinuities, we combine the high- and low-pass filter parts using a parameter μ, which makes our model applicable to different data.

$$\bar{X}=[(I-\frac{L}{2})^k+\mu(\frac{L}{2})^k]X. \tag{8}$$

According to Proposition 1, performing graph filtering in this way allows us to obtain information from surrounding nodes while ensuring the uniqueness of each node. Thus, the smoothed $\bar{X}$ provides a good data representation for downstream task.

Proposition 1. *The proposed adaptive filter can make features more discriminative than a low-pass filter. A high-pass filter increases the Dirichlet energy.*

Proof. For two connected nodes x_i and x_j, let T, T_L, T_H and T_A be the distances between them obtained through the original graph, low-pass filter, high-pass filter, adaptive filter, respectively. After filtering by Eq. (5) once, a node's feature becomes:

$$\bar{X}_{i,\cdot}=\left(\frac{1}{2}+\frac{\widetilde{a}_{ii}}{2}\right)x_i+\sum_{p\neq i}\frac{\widetilde{a}_{ip}}{2}x_p$$

Similarly, filtering with Eq. (7) yields:

$$\bar{X}_{i,\cdot} = \left(\frac{1}{2} - \frac{\widetilde{a}_{ii}}{2}\right) x_i - \sum_{p \neq i} \frac{\widetilde{a}_{ip}}{2} x_p$$

Therefore, both low-pass and high-pass filters aggregate features linearly by summarization or subtraction. Note that a node's feature plays a more important role than its neighbors in low-pass filtering, whereas it is the opposite in high-pass filtering. We use constants ε_1, $\varepsilon_2 \in (0, 1)$ to indicate this effect. Then, we obtain:

$$T = \left\| x_i - x_j \right\|_2$$

$$T_L = \left\| \left(x_i + \varepsilon_1 x_j\right) - \left(x_j + \varepsilon_1 x_i\right) \right\|_2 = \left(1 - \varepsilon_1\right) T$$

$$T_H = \left\| \left(\varepsilon_2 x_i - x_j\right) - \left(\varepsilon_2 x_j - x_i\right) \right\|_2 = \left(1 + \varepsilon_2\right) T$$

$$T_A = T_L + \mu T_H = \left[\left(1 - \varepsilon_1\right) + \mu \left(1 + \varepsilon_2\right)\right] T$$

Notably, $T_A > T_L$, which means that the adaptive filter can further push neighbors away w.r.t. the low-pass filter. Therefore, this could alleviate the over-smoothing issue to some extent. When $\mu > \frac{\varepsilon_1}{1+\varepsilon_2}$, we obtain $T_A > T$. This indicates that an adaptive filter can make the original nodes apart and the upper limit is bounded by the parameter μ. This improves feature discrimination.

Similarly, let D, D_L, D_H and D_A be the Dirichlet energy of raw data, filtered by low-pass filter, high-pass filter, and adaptive filter. We have:

$$D = \frac{1}{2} \sum_{i,j \in E} \widetilde{A}_{ij} \left\| x_i - x_j \right\|_2^2$$

$$D_L = \frac{1}{2} \sum_{i,j \in E} \widetilde{A}_{ij} \left\| \left(x_i + \varepsilon_1 x_j\right) - \left(x_j + \varepsilon_1 x_i\right) \right\|_2^2$$
$$= (1 - \varepsilon_1)^2 D$$

$$D_H = \frac{1}{2} \sum_{i,j \in E} \widetilde{A}_{ij} \left\| \left(\varepsilon_2 x_i - x_j\right) - \left(\varepsilon_2 x_j - x_i\right) \right\|_2^2$$
$$= (1 + \varepsilon_2)^2 D$$

$$D_A = D_L + \mu D_H = (1 - \varepsilon_1)^2 D + \mu (1 + \varepsilon_2)^2 D$$

Here, $D_A > D_L$, i.e., adaptive filter increases the Dirichlet energy of low-pass filter.

Based on Proposition 1, adaptive filter can bring homophilic nodes closer while pushing heterophilic nodes further apart, which improves downstream task performance.

3.3. Graph learning

The original graph A is typically noisy and sparse, which degrades the performance when directly applied to downstream task. To circumvent this problem, we learn a refined graph from the raw data. Our methods stem from the self-expression property of data, where each data sample can be represented by a linear combination of other samples in the same group (Lu et al., 2012; Lv, Kang, Lu, & Xu, 2021). The objective function to compute the refined graph S from $\bar{X}$ can be mathematically formulated as follows:

$$\min_S \left\| \bar{X}^\top - \bar{X}^\top S \right\|_F^2 + \beta \left\| S \right\|_F^2, \tag{9}$$

where $S \in \mathcal{R}^{N \times N}$ and $\beta > 0$ represents a parameter. The first term acts as a reconstruction loss and the second term is a regularization term. This simple regularizer fails to fully mine the latent information in the graph. Inspired by the success of contrastive learning, we use a similar idea to improve graph quality. Most graph contrastive learning methods are performed at node-level and positive pairs are constructed by data augmentation, which could incur negative effect (Trivedi, Lubana, Yan, Yang, & Koutra, 2022). Different from them, we employ a contrastive regularizer at graph-level, which does not assume a predefined graph. In particular, each node is considered the anchor, and its K-nearest neighbors (KNN) are regarded as positive pairs, whereas the remaining nodes are regarded as negative samples. This can be expressed as follows:

$$\mathcal{J} = -\sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} \log \frac{\exp\left(S_{ij}\right)}{\sum_{p \neq i}^{N} \exp\left(S_{ip}\right)}. \tag{10}$$

Eq. (10) brings similar samples closer and pushes dissimilar samples further apart to improve the discriminability of the learned graph. Specifically, similar nodes will have close representations and large edge weights, whereas dissimilar ones will be distinct and have small or even edge weights of zero. To sum up, our proposed Contrastive Graph Clustering (CGC) model can be formulated as follows:

$$\min_S \left(\left\| \bar{X}^\top - \bar{X}^\top S \right\|_F^2 + \beta \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} -\log \frac{\exp\left(S_{ij}\right)}{\sum_{p \neq i}^{N} \exp\left(S_{ip}\right)} \right). \tag{11}$$

3.4. Optimization

To update S, we adopt a gradient descent strategy. Its derivative at epoch t can be abbreviated as follows:

$$\nabla_1^{(t)} + \beta \nabla_2^{(t)}. \tag{12}$$

The first term is

$$\nabla_1^{(t)} = 2\left(-\left[\bar{X}\bar{X}^\top\right]_{ij} + \left[\bar{X}\bar{X}^\top S^{(t-1)}\right]_{ij}\right). \tag{13}$$

Define $M^{(\mathrm{t}-1)} = \sum_{p \neq i}^{N} \exp\left(S_{ip}^{(t-1)}\right)$ and let n be the total number of neighbors. We generally set n to 10. Consequently, the second term becomes

$$\nabla_2^{(t)} = \begin{cases} -1 + \frac{n \exp\left(S_{ij}^{(t-1)}\right)}{M^{(t-1)}}, & \text{if } j \text{ in } \mathcal{N}_i \\ \frac{n \exp\left(S_{ij}^{(t-1)}\right)}{M^{(t-1)}}, & \text{otherwise} \end{cases} \tag{14}$$

Then Adam optimization is applied to optimize S. S is updated until convergence. Algorithm 1 summarizes the entire process. Our method's computational complexity is $O(N^2)$, whereas relevant methods FGC is $O(N^3)$ and AGC is $O(N^2 d)$. The implementation is public available at: https://github.com/XieXuanting/CGC.

Algorithm 1 CGC

```
Require: adjacency matrix $A$, feature $X$, the order of graph filtering $k$,
    parameters $\mu$ and $\beta$, the number of clusters $c$. $D = (A + I)\mathbf{1}_N$.
Ensure: $c$ clusters
$\widetilde{A} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}$
$L = I - \widetilde{A}$
Graph filtering by Eq. (8)
while convergence condition does not meet do
    Update $S$ in Eq. (11) via Adam
end while
$C = \frac{(|S| + |S|^\top)}{2}$
Clustering on $C$
```

4. Experiments

4.1. Datasets

We test our methods on various benchmark datasets that are typically used to evaluate the performance of graph analysis techniques. Cora, Citeseer, and Pubmed are citation networks (Kipf & Welling, 2016). These nodes and edges show the citation and cited relationships among the papers. Wiki is a webpage network (Wang, Jin, Cao, Yang, & Zhang, 2016). The nodes in this network represent web pages and are connected if one links to another. The features in Cora and Citeseer

Table 1
The statistics of all datasets.

Dataset	Nodes	Edges	Features	Classes	Homophily	Density
Cora	2 708	5 429	1433	7	0.83	0.15%
Citeseer	3 327	4 732	3703	6	0.71	0.08%
Pubmed	19 717	44 338	500	3	0.79	0.02%
Wiki	2 405	17 981	4973	17	0.46	0.62%
Large Cora	11 881	64 898	3780	10	0.73	0.09%
Squirrel	5,201	217,073	2089	5	0.22	1.60%
Wisconsin	251	515	1703	5	0.16	1.63%
Cornell	183	298	1703	5	0.11	1.78%
Texas	183	325	1703	5	0.06	1.94%

are binary word vectors. Pubmed and Wiki are characterized by tf-idf weighted word vectors. In addition, we add another large dataset, namely, Large Cora (Li, Wu, Liu, Zhang, & Guan, 2019), which has the most edges. To further test our model in real-world applications, we evaluate the proposed model in clustering task on heterophilic graph datasets, including webgraphs from WebKB datasets[1]: WISCONSIN, CORNELL and TEXAS, and Wikipedia networks SQUIRREL (Rozemberczki, Allen, & Sarkar, 2021). To compute the graph's homophily, we use an index of homophily introduced by Pei, Wei, Chang, Lei, and Yang (2020) as follows:

$$homophily = \frac{1}{N}\sum_{v\in\mathcal{V}} \alpha_v \ \ and \ \ \alpha_v = \frac{\left|\{u \in \mathcal{N}_v | \ell(u) = \ell(v)\}\right|}{|\mathcal{N}_v|}, \tag{15}$$

where $\ell(v)$ indicates the label of node v. A large homophily value means strong homophily in a graph. We also calculate the density of the original graph A, which is the ratio of element 1. The statistics of these datasets in detail are summarized in Table 1.

4.2. Baselines

We compare our method CGC with many popular algorithms. These include (1) clustering methods that rely only on the graph structure, including Spectral-g (spectral clustering that uses only the structural information of the graph), modularized nonnegative matrix factorization (M-NMF) based network embedding (Wang et al., 2017), DeepWalk (Perozzi et al., 2014), and deep neural networks for graph representations (DNGR) (Cao et al., 2016). (2) Clustering methods that only use the attributes of nodes, including k-means and Spectral-f (Spectral clustering, which only uses information from node features to construct graph matrix by linear kernel). (3) Attributed graph clustering methods that make full use of both node features and topological structure information, including SCI (Wang et al., 2016), ARGE and ARVGE (Pan et al., 2018), GAE and VGAE (Kipf & Welling, 2016), MGAE (Wang, Pan et al., 2017), AGC (Zhang et al., 2019), DAEGC (Wang et al., 2019), SDCN (Bo et al., 2020), ARGA_AX and ARVGA_AX (Pan et al., 2019), DGI (Veličković et al., 2019), VGAE with Gaussian mixture models (GMM-VGAE) (Hui et al., 2020). EVGC (Guo & Dai, 2021), DNENC-Att (with graph attentional autoencoder) and DNENC-Con (with graph convolutional autoencoder) (Wang et al., 2022), ARVGA-Col-M and RWR-Col-M (Zhu et al., 2022), FGC (Kang et al., 2022) are the most recent methods.

4.3. Setup

We set the maximum number of iterations to 30 and finish the training when the loss decreases slowly. We find the stop epoch for the datasets as follows: 22 on Cora, 23 on Citeseer, 30 on Wiki, 26 on Pubmed and Large Cora, 25 on Squirrel and 7 on Wisconsin, Cornell and Texas. The learning rate is set to 0.004 for Citeseer and 0.0001 for other datasets. To evaluate the performance of different methods, the metrics we adopt include (1) clustering accuracy (ACC), which is used to evaluate the performance of label matching clustering results. (2) Normalized mutual information (NMI), which calculates the mutual information entropy between the ground truth and cluster labels. (3) Macro F1-score (F1), which is the harmonic mean of recall and precision. The higher aforementioned metric values, the better the method. We perform grid search to find the best hyperparameters. For a fair comparison, we use the settings in AGC, and directly quote some parts of the results from AGC, DAEGC, etc. For the heterophilic graph datasets, DAEGC and FGC are tested with the provided code. The results of the other methods, which are the most recent deep methods, are taken directly from the Zhu et al. (2022).

4.4. Result analysis

Table 2 summarizes the experiment outcomes on homophilic datasets and Table 3 summarizes the results on heterophilic graphs, with the best results highlighted in bold. The proposed method CGC outperforms the other state-of-the-art methods on Cora, Citeseer, Wiki, Large Cora, Squirrel, Wisconsin, Cornell, and Texas. However, on Pubmed, our method's performance is slightly poor. This could be because the graph is very sparse, which makes it difficult to select appropriate positive pairs. The detailed descriptions are as follows:

1. CGC can significantly outperform state-of-the-art baseline FGC in most cases. FGC is also without neural networks. The first step of FGC is filtering with a low-pass filter. The results demonstrate the effectiveness of our proposed adaptive filter.
2. CGC outperforms the recent deep learning methods, i.e., EVGC, DNENC-Att, DNENC-Con, ARVGA-Col-M and RWR-Col-M in most cases. Note that they are all deep learning methods, and thus our simple approach has a significant advantage in practice.
3. CGC outperforms the other GCN-based clustering methods in most cases, including SCI, GAE, VGAE, MGAE, AGC, ARGE, ARVGE, DAEGC, SDCN, ARGA_AX, ARVGA_AX, and GMM-VGAE. In particular, on Wiki and Large Cora, our approach achieves high performance. Most of the other GCN-based clustering methods learn latent representations for spectral clustering, whereas our method learns a refined graph and uses the learned graph directly for downstream tasks. The results verify that a refined graph can significantly improve the clustering performance.
4. CGC significantly outperforms the contrastive learning method DGI. Furthermore, with respect to complex contrastive learning methods, our approach is easy to understand and simple to implement.
5. Methods that use both topology and attribute information typically outperform those that exploit only one type of information. This validates the importance of developing methods that explore both attribute and structure information.
6. CGC achieves the best performance on all heterophilic graph datasets. These results suggest that CGC generalizes well on different types of graphs, verifying that our adaptive filter is suitable for real-world data.

5. Ablation study

5.1. Contrastive regularizer

To assess the impact of the contrastive regularizer, we replace the contrastive loss $\mathcal{J}$ with a Frobenius term. The total loss function is Eq. (9) and it is marked as CGC w/o $\mathcal{J}$. Table 4 shows the performance degradation on all datasets. With the contrastive regularizer, CGC's ACC is increased by 4.54%, 2.34%, 0.93%, 2.74% and 6.72% on Cora, Citeseer, Pubmed, Wiki and Large Cora, respectively. We see similar improvements in other metrics.

[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/.

Table 2
Clustering performance on homophilic datasets.

Methods	Input	Cora			Citeseer			Pubmed			Wiki			Large Cora		
		ACC%	NMI%	F1%	ACC%	NMI%	F1%	ACC%	NMI%	F1%	ACC%	NMI%	F1%	ACC%	NMI%	F1%
Spectral-g	Graph	34.19	19.49	30.17	25.91	11.84	29.48	39.74	3.46	51.97	23.58	19.28	17.21	39.84	8.24	10.70
DNGR (Cao, Lu, & Xu, 2016)	Graph	49.24	37.29	37.29	32.59	18.02	44.19	45.35	15.38	17.90	37.58	35.85	25.38	–	–	–
DeepWalk (Perozzi, Al-Rfou, & Skiena, 2014)	Graph	46.74	31.75	38.06	36.15	9.66	26.70	61.86	16.71	47.06	38.46	32.38	25.74	–	–	–
M-NMF (Wang et al., 2017)	Graph	42.30	25.60	32.00	33.60	9.90	25.50	47.00	8.40	44.30	–	–	–	–	–	–
K-means	Feature	34.65	16.73	25.42	38.49	17.02	30.47	57.32	29.12	57.35	33.37	30.20	24.51	33.09	9.36	11.31
Spectral-f	Feature	36.26	15.09	25.64	46.23	21.19	33.70	59.91	32.55	58.61	41.28	43.99	25.20	29.71	11.65	17.76
SCI (Wang et al., 2016)	Both	41.21	21.57	11.82	33.45	9.77	18.01	44.89	5.99	35.73	32.72	26.38	19.03	26.78	11.31	7.68
ARGE (Pan, Hu, Long, Jiang, Yao, & Zhang, 2018)	Both	64.00	44.90	61.90	57.30	35.00	54.60	59.12	23.17	58.41	41.40	39.50	38.27	–	–	–
ARVGE (Pan et al., 2018)	Both	63.80	45.00	62.70	54.40	26.10	52.90	58.22	20.62	23.04	41.55	40.01	37.80	–	–	–
GAE (Kipf & Welling, 2016)	Both	53.25	40.69	41.97	41.26	18.34	29.13	64.08	22.97	49.26	17.33	11.93	15.35	–	–	–
VGAE (Kipf & Welling, 2016)	Both	55.95	38.45	41.50	44.38	22.71	31.88	65.48	25.09	50.95	28.67	30.28	20.49	–	–	–
MGAE (Wang, Pan et al., 2017)	Both	63.43	45.57	38.01	63.56	39.75	39.49	43.88	8.16	41.98	50.14	47.97	39.20	38.04	32.43	29.02
AGC (Zhang et al., 2019)	Both	68.92	53.68	65.61	67.00	41.13	62.48	69.78	31.59	68.72	47.65	45.28	40.36	40.54	32.46	31.84
DAEGC (Wang, Pan, Hu, Long, Jiang, & Zhang, 2019)	Both	70.40	52.80	68.20	67.20	39.70	63.60	67.10	26.60	65.90	38.25	37.63	23.64	39.87	32.81	19.05
SDCN (Bo et al., 2020)	Both	60.24	50.04	61.84	65.96	38.71	63.62	65.78	29.47	65.16	–	–	–	–	–	–
ARGA_AX (Pan et al., 2019)	Both	59.70	45.50	57.90	54.70	26.30	52.70	63.70	24.50	63.90	–	–	–	–	–	–
ARVGA_AX (Pan et al., 2019)	Both	71.10	52.60	69.30	58.10	33.80	52.50	64.00	23.90	64.40	–	–	–	–	–	–
DGI (Veličković et al., 2019)	Both	71.81	54.09	69.88	68.60	43.75	64.64	–	–	–	44.37	42.20	40.16	–	–	–
GMM-VGAE (Hui et al., 2020)	Both	71.50	54.43	67.76	67.44	42.30	63.22	71.03	30.28	69.74	–	–	–	–	–	–
EVGC (Guo & Dai, 2021)	Both	72.95	55.76	71.01	67.02	41.89	62.89	70.80	**35.63**	70.32	51.46	49.37	45.14	–	–	–
DNENC-Att (Wang et al., 2022)	Both	70.40	52.80	68.20	67.20	39.70	63.60	67.10	26.60	65.90	–	–	–	–	–	–
DNENC-Con (Wang et al., 2022)	Both	68.30	51.20	65.90	69.20	42.60	63.90	67.70	27.50	67.50	–	–	–	–	–	–
ARVGA-Col-M (Zhu, Li, Wang, Xiao, Zhao, & Hu, 2022)	Both	69.31	51.93	66.45	59.51	31.59	57.04	69.98	29.78	68.76	–	–	–	–	–	–
RWR-Col-M (Zhu et al., 2022)	Both	71.64	53.38	**71.11**	64.56	37.71	61.24	**76.97**	40.38	**76.59**	–	–	–	–	–	–
FGC (Kang et al., 2022)	Both	72.90	56.12	63.27	69.01	**44.02**	64.43	70.01	31.56	69.10	51.10	44.12	34.79	48.25	**35.24**	35.52
CGC	Both	**75.15**	**56.90**	66.22	**69.31**	43.61	**64.74**	67.43	33.07	67.14	**59.04**	**53.20**	**45.43**	**50.18**	34.10	**43.79**

Table 3
Clustering performance on heterophilic graphs.

Methods	Input	Squirrel			Wisconsin			Cornell			Texas		
		ACC%	NMI%	F1%	ACC%	NMI%	F1%	ACC%	NMI%	F1%	ACC%	NMI%	F1%
DAEGC (Wang et al., 2019)	Both	25.55	2.36	**24.07**	39.62	12.02	6.22	42.56	12.37	30.20	45.99	11.25	18.09
ARVGA-Col-M (Zhu et al., 2022)	Both	–	–	–	54.34	11.41	–	–	–	–	59.89	16.37	–
RWR-Col-M (Zhu et al., 2022)	Both	–	–	–	53.58	16.25	–	–	–	–	57.22	13.82	–
FGC (Kang et al., 2022)	Both	25.11	1.32	22.13	50.19	12.92	25.93	44.10	8.60	**32.68**	53.48	5.16	17.04
CGC	Both	**27.23**	**2.98**	20.57	**55.85**	**23.03**	**27.29**	**44.62**	**14.11**	21.91	**61.50**	**21.48**	**27.20**

Table 4
Results of CGC w/o $\mathcal{J}$, without filter CGC−, only with low-pass filter CGC+, and with initial node features CGC*.

Methods	Cora			Citeseer			Pubmed			Wiki			Large Cora		
	ACC%	NMI%	F1%	ACC%	NMI%	F1%	ACC%	NMI%	F1%	ACC%	NMI%	F1%	ACC%	NMI%	F1%
CGC w/o $\mathcal{J}$	70.61	49.30	62.08	66.97	41.77	61.11	66.50	30.89	65.95	56.30	52.43	44.84	43.46	27.49	31.72
CGC−	72.93	56.40	65.20	58.34	37.81	55.01	65.81	32.31	65.78	55.34	50.52	38.04	45.94	32.82	42.27
CGC+	74.15	58.46	66.10	64.95	40.35	59.83	66.95	32.45	66.91	57.34	54.30	42.21	49.37	20.74	26.33
CGC*	75.07	**59.74**	**66.75**	66.73	41.10	61.12	67.05	30.55	66.92	58.59	54.59	42.14	46.23	32.81	42.51
CGC	**75.15**	56.90	66.22	**69.31**	**43.61**	**64.74**	**67.43**	**33.07**	**67.14**	**59.04**	**53.20**	**45.43**	**50.18**	**34.10**	**43.79**

5.2. Graph filtering

To understand the contribution of graph filtering, we conduct a series of experiments. The method without filtering is marked as CGC−. We also replace the filter only with low-pass filtering, i.e., Eq. (5), which is dubbed as CGC+. Note that CGC+ becomes the same as MCGC (Pan & Kang, 2021) in a single-view setting. As shown in Table 4, for CGC without filtering, the ACC on Cora, Citeseer, Pubmed, Wiki and Large Cora decreases by 2.22%, 10.97%, 1.62%, 3.70%, and 4.24%, respectively. This verifies the effectiveness of graph filtering. For CGC without high-pass filter, the ACC on Cora, Citeseer, Pubmed, Wiki and Large Cora decreases by 1.00%, 4.36%, 0.48%, 1.70%, and 0.81%, respectively. This validates the benefit of adaptive filter.

Besides, we design another scenario to determine the effect of high-pass filter. Rather than combining low- and high-pass information, we adaptively combine low-pass information with the initial feature, i.e.,

$$\bar{X} = (I - \frac{L}{2})^k X + \mu X. \tag{16}$$

Filtering in this way is marked as CGC* in Table 4. Our method still outperforms this approach. Moreover, CGC* outperforms CGC+ in most cases. This suggests that it is not advisable to simply use low-pass filtering because a low-pass filter removes some useful information. In summary, applying an adaptive filter such as Eq. (8) is the best choice in practice.

5.3. Parameter analysis

The filter has two parameters, filter order k and trade-off parameter μ. We set $k = [1, 2, 3, 4, 5]$ and $\mu = [0.01, 0.1, 1, 10, 100]$. Their influence on clustering performance on Cora and Large Cora is shown in Fig. 2. Our method performs well for a wide range of parameters. When k increases, nearby node features become similar. However, too large k would lead to over-smoothing. CGC yields a reasonable outcome for a small k. A large μ degrades the performance because too much noise is incorporated.

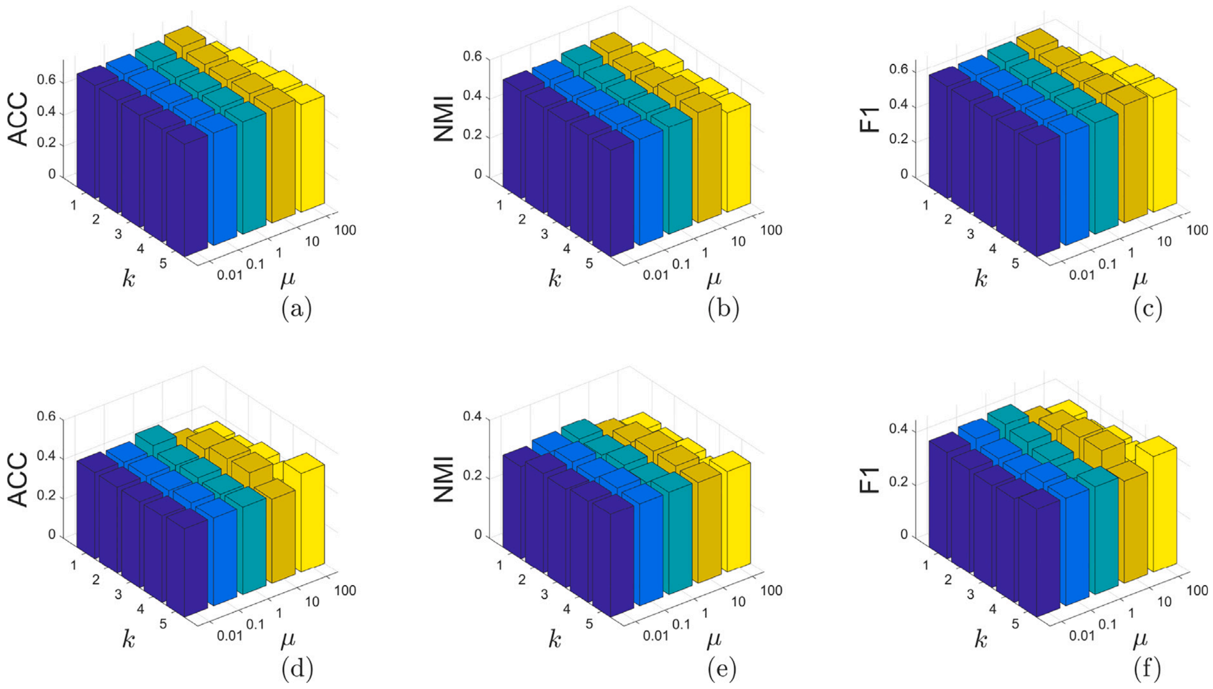

Fig. 2. Sensitivity analysis of parameters k and μ on Cora (a-c) and Large Cora (d-f).

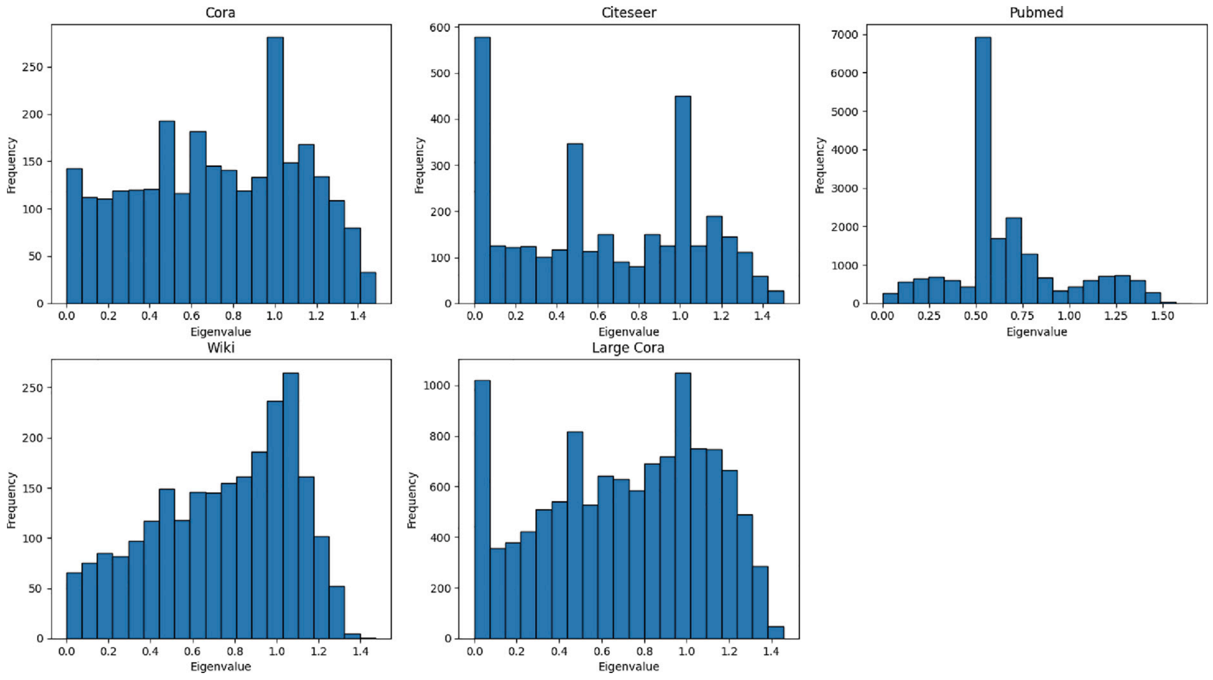

Fig. 3. The eigenvalue distributions of benchmark datasets.

The value of μ is significantly determined by the specific data because different data contain various levels of high-frequency information. To demonstrate this, we show the distribution of eigenvalues in Fig. 3. Take Citeseer and Large Cora as examples, the former's eigenvalues of more than 0.6 account for 51.28%, whereas the latter is 60.17%. Thus, we set $\mu = 2$ on Citeseer and $\mu = 100$ on Large Cora because Large Cora should retain more high-frequency information.

To observe the impact of the balance parameter β and neighbor n on graph learning, we tune $\beta = [0.01, 0.1, 1, 10, 100, 1000]$ and $n = [1, 10, 100, 1000, 10\,000]$ on Large Cora. As illustrated in Fig. 4, a small n is preferred for most β values. We fix n to 10 on all datasets. Besides, we plot the objective loss's variation of Eq. (11) on homophilic dataset Wiki and heterophilic dataset Texas in Fig. 5. The figure shows that the objective loss has a clear trend toward convergence.

6. Conclusion

In this study, we propose a graph clustering method based on adaptive filter. This new filter, which combines low-pass and high-pass filtering, captures the entire spectral information and does not assume

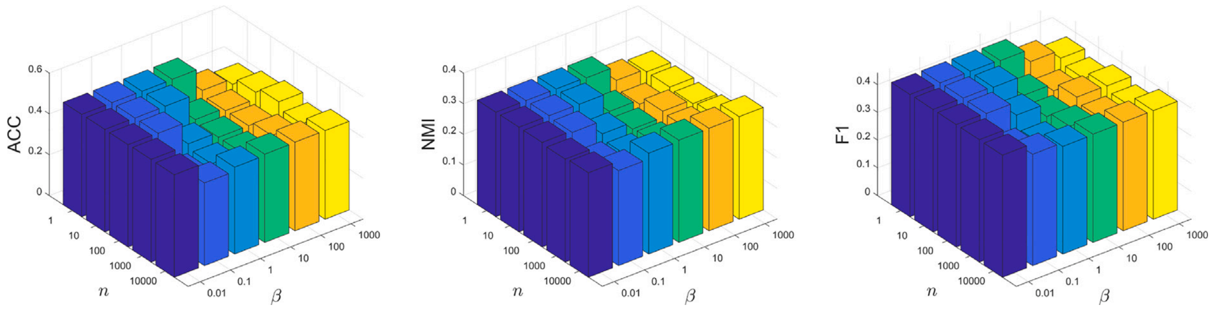

Fig. 4. Sensitivity analysis of parameters β and n on Large Cora.

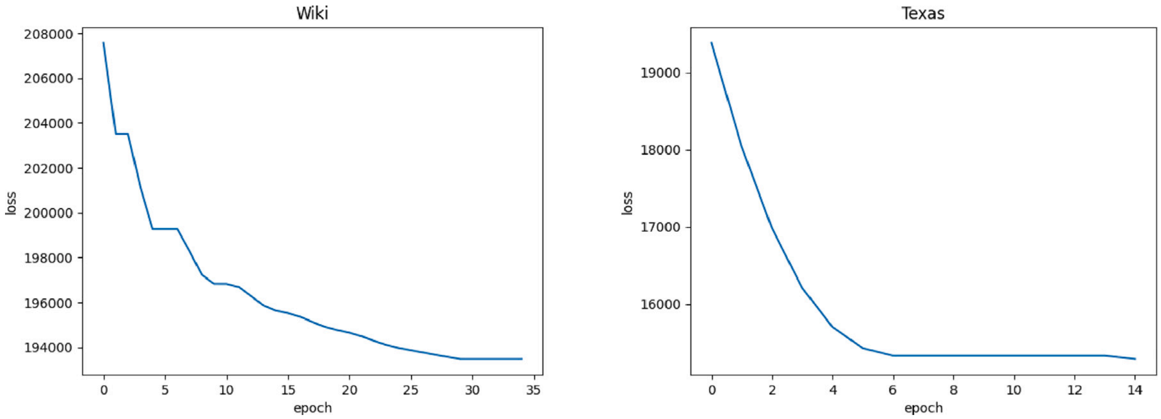

Fig. 5. The evolution of objective function on Wiki and Texas.

textcolorbluethe homophily of data; thus it is attractive in practice. With the help of this filter, node feature information is enriched and a refined graph is learned upon the smoothed data representation. Our experimental results show that CGC achieves significant improvements over the baseline methods on graph clustering task. When faced with the systematic usage of complicated deep neural networks, this study shows that shallow models can still outperform deep learning approaches in some applications. One potential limitation of our method is that it can consume a significant amount of memory due to the learned graph's $N \times N$ size. This problem can be alleviated to some extent by implementing it in parallel because our method just involves gradient computation. Future studies will focus on scalability.

CRediT authorship contribution statement

Xuanting Xie: Methodology, Software, Validation, Formal analysis, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Wenyu Chen:** Validation, Formal analysis, Data curation, Writing – review & editing. **Zhao Kang:** Conceptualization, Investigation, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Chong Peng:** Validation, Writing – review & editing.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 62276053).

References

Balcilar, M., Renton, G., Héroux, P., Gaüzère, B., Adam, S., & Honeine, P. (2020). Analyzing the expressive power of graph neural networks in a spectral perspective. In *International conference on learning representations*.

Bo, D., Wang, X., Shi, C., & Shen, H. (2021). Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 3950–3957).

Bo, D., Wang, X., Shi, C., Zhu, M., Lu, E., & Cui, P. (2020). Structural deep clustering network. In *Proceedings of the web conference 2020* (pp. 1400–1410).

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2014). Spectral networks and deep locally connected networks on graphs. In *2nd international conference on learning representations, ICLR 2014*.

Cao, S., Lu, W., & Xu, Q. (2016). Deep neural networks for learning graph representations. In *Proceedings of the AAAI conference on artificial intelligence*.

Chang, H., Rong, Y., Xu, T., Huang, W., Sojoudi, S., Huang, J., et al. (2021). Spectral graph attention network with fast eigen-approximation. In *Proceedings of the 30th ACM international conference on information & knowledge management* (pp. 2905–2909).

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597–1607). PMLR.

Chen, Y., Wu, L., & Zaki, M. (2020). Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in Neural Information Processing Systems*, *33*, 19314–19326.

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, *29*, 3844–3852.

Dong, X., Thanou, D., Rabbat, M., & Frossard, P. (2019). Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, *36*(3), 44–63.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, *33*, 21271–21284.

Guo, L., & Dai, Q. (2021). End-to-end variational graph clustering with local structural preservation. *Neural Computing and Applications*, 1–16.

Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)* (pp. 1735–1742). IEEE.

Hassani, K., & Khasahmadi, A. H. (2020). Contrastive multi-view representation learning on graphs. In *International conference on machine learning* (pp. 4116–4126). PMLR.

Henaff, M., Bruna, J., & LeCun, Y. (2015). Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163.

Huang, S., Kang, Z., Xu, Z., & Liu, Q. (2021). Robust deep k-means: An effective and simple method for data clustering. *Pattern Recognition*, *117*, Article 107996.

Hui, B., et al. (2020). Collaborative graph convolutional networks: Unsupervised learning meets semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 4215–4222).

Jin, D., Yu, Z., Huo, C., Wang, R., Wang, X., He, D., et al. (2021). Universal graph convolutional networks. *Advances in Neural Information Processing Systems*, *34*.

Kang, Z., et al. (2022). Fine-grained attributed graph clustering. In *Proceedings of the 2022 SIAM international conference on data mining* (pp. 370–378). SIAM.

Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. In *NIPS bayesian deep learning workshop*.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *ICLR*.

Klicpera, J., Bojchevski, A., & Günnemann, S. (2018). Predict then propagate: Graph neural networks meet personalized PageRank. In *International conference on learning representations*.

Li, Y., Hu, P., Liu, Z., Peng, D., Zhou, J. T., & Peng, X. (2021). Contrastive clustering. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 8547–8555).

Li, S., Kim, D., & Wang, Q. (2021). Beyond low-pass filters: Adaptive feature propagation on graphs. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 450–465). Springer.

Li, Q., Wu, X.-M., Liu, H., Zhang, X., & Guan, Z. (2019). Label efficient semi-supervised learning via graph filtering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9582–9591).

Lin, Z., Kang, Z., Zhang, L., & Tian, L. (2023). Multi-view attributed graph clustering. *IEEE Transactions on Knowledge and Data Engineering*, *35*(2), 1872–1880.

Liu, L., Chen, P., Luo, G., Kang, Z., Luo, Y., & Han, S. (2022). Scalable multi-view clustering with graph filtering. *Neural Computing and Applications*, *34*(19), 16213–16221.

Liu, Y., Tu, W., Zhou, S., Liu, X., Song, L., Yang, X., et al. (2022). Deep graph clustering via dual correlation reduction. In *Proc. of AAAI*.

Liu, C., Wen, L., Kang, Z., Luo, G., & Tian, L. (2021). Self-supervised consensus representation learning for attributed graph. In *Proceedings of the 29th ACM international conference on multimedia* (pp. 2654–2662).

Lu, C.-Y., Min, H., Zhao, Z.-Q., Zhu, L., Huang, D.-S., & Yan, S. (2012). Robust and efficient subspace segmentation via least squares regression. In *European conference on computer vision* (pp. 347–360). Springer.

Lv, J., Kang, Z., Lu, X., & Xu, Z. (2021). Pseudo-supervised deep subspace clustering. *IEEE Transactions on Image Processing*, *30*, 5252–5263.

Ma, Z., Kang, Z., Luo, G., Tian, L., & Chen, W. (2020). Towards clustering-friendly representations: Subspace clustering via graph filtering. In *Proceedings of the 28th ACM international conference on multimedia* (pp. 3081–3089).

Murtagh, F., & Contreras, P. (2012). Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*, 86–97.

Pan, S., Hu, R., Fung, S.-f., Long, G., Jiang, J., & Zhang, C. (2019). Learning graph embedding with adversarial training methods. *IEEE Transactions on Cybernetics*, *50*(6), 2475–2487.

Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., & Zhang, C. (2018). Adversarially regularized graph autoencoder for graph embedding. In *IJCAI*.

Pan, E., & Kang, Z. (2021). Multi-view contrastive graph clustering. *Advances in Neural Information Processing Systems*, *34*.

Pandit, S., Chau, D. H., Wang, S., & Faloutsos, C. (2007). Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on world wide web* (pp. 201–210).

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., & Yang, B. (2020). Geom-GCN: Geometric graph convolutional networks. In *International conference on learning representations*.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 701–710).

Rozemberczki, B., Allen, C., & Sarkar, R. (2021). Multi-scale attributed node embedding. *Journal of Complex Networks*, *9*(1), 1–22.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., & Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, *30*, 83–98.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web* (pp. 1067–1077).

Tian, F., Gao, B., Cui, Q., Chen, E., & Liu, T.-Y. (2014). Learning deep representations for graph clustering. In *Proceedings of the AAAI conference on artificial intelligence*.

Trivedi, P., Lubana, E. S., Yan, Y., Yang, Y., & Koutra, D. (2022). Augmentations in graph contrastive learning: Current methodological flaws & towards better practices. In *Proceedings of the 2022 world wide web conference*.

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., & Hjelm, R. D. (2019). Deep graph infomax. In *ICLR*.

Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., & Yang, S. (2017). Community preserving network embedding. In *Thirty-First AAAI conference on artificial intelligence*.

Wang, X., Jin, D., Cao, X., Yang, L., & Zhang, W. (2016). Semantic community identification in large attribute networks. In *Proceedings of the AAAI conference on artificial intelligence*.

Wang, C., Pan, S., Celina, P. Y., Hu, R., Long, G., & Zhang, C. (2022). Deep neighbor-aware embedding for node clustering in attributed graphs. *Pattern Recognition*, *122*, Article 108230.

Wang, C., Pan, S., Hu, R., Long, G., Jiang, J., & Zhang, C. (2019). Attributed graph clustering: A deep attentional embedding approach. In *IJCAI*.

Wang, C., Pan, S., Long, G., Zhu, X., & Jiang, J. (2017). Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 889–898).

Wu, Z., Pan, S., Long, G., Jiang, J., & Zhang, C. (2022). Beyond low-pass filtering: Graph convolutional networks with automatic filtering. *IEEE Transactions on Knowledge and Data Engineering*.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019). Simplifying graph convolutional networks. In *International conference on machine learning* (pp. 6861–6871). PMLR.

Xia, W., Gao, Q., Yang, M., & Gao, X. (2021). Self-supervised contrastive attributed graph clustering. arXiv preprint arXiv:2110.08264.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., & Shen, Y. (2020). Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, *33*, 5812–5823.

Zhang, X., Liu, H., Li, Q., & Wu, X.-M. (2019). Attributed graph clustering via adaptive graph convolution. In *IJCAI*.

Zhu, H., & Koniusz, P. (2020). Simple spectral graph convolution. In *International conference on learning representations*.

Zhu, P., Li, J., Wang, Y., Xiao, B., Zhao, S., & Hu, Q. (2022). Collaborative decision-reinforced self-supervision for attributed graph clustering. *IEEE Transactions on Neural Networks and Learning Systems*.

Zhu, J., Rossi, R. A., Rao, A., Mai, T., Lipka, N., Ahmed, N. K., et al. (2021). Graph neural networks with heterophily. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 11168–11176).

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., & Koutra, D. (2020). Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, *33*, 7793–7804.