Brief Papers

Seeing All From a Few: Nodes Selection Using Graph Pooling for Graph Clustering

Yiming Wang†, Dongxia Chang‡, Zhiqiang Fu†, and Yao Zhao‡, Senior Member, IEEE

Abstract—Recently, there has been considerable research interest in graph clustering aimed at data partition using graph information. However, one limitation of most graph-based methods is that they assume that the graph structure to operate is reliable. However, there are inevitably some edges in the graph that are not conducive to graph clustering, which we call spurious edges. This brief is the first attempt to employ the graph pooling technique for node clustering to the best of our knowledge. In this brief, we propose a novel dual graph embedding network (DGEN), which is designed as a two-step graph encoder connected by a graph pooling layer to learn the graph embedding. In DGEN, we assume that if a node and its nearest neighboring node are close to the same clustering center, this node is informative, and this edge can be considered as a cluster-friendly edge. Based on this assumption, the neighbor cluster pooling (NCPool) is devised to select the most informative subset of nodes and the corresponding edges based on the distance of nodes and their nearest neighbors to the cluster centers. This can effectively alleviate the impact of the spurious edges on the clustering. Finally, to obtain the clustering assignment of all nodes, a classifier is trained using the clustering results of the selected nodes. Experiments on five benchmark graph datasets demonstrate the superiority of the proposed method over state-of-the-art algorithms.

Index Terms—Clustering, graph neural networks (GNNs), graph pooling.

I. INTRODUCTION

With the advance of information technology, non-Euclidean domain data can be readily acquired in many domains and have great value for exploitation, such as human pose [1], protein structure [2], and social networks [3]. As one of the most fundamental tasks, clustering divides these non-Euclidean data samples according to their similarity, which is called the graph clustering method [4], [5]. These conventional methods, however, proved to be incomplete in mining nodes relationship for their over-reliance on prior knowledge.

To mine non-Euclidean graph information, researchers have devised graph neural networks (GNNs) [6], [7] capable of encoding both graph structure and node characteristics for node latent representation. GNNs have successfully expanded deep learning techniques to non-Euclidean graph data with remarkable achievement made in multiple graph tasks, such as medical image classification [8], 3-D object detection [9], and visual question answering [10]. Representatively, de Vriendt et al. [8] first propose to employ a deep graph framework for semi-supervised medical image classification and obtain the desired performance. Because of the properties of graph convolution, many GNN-based graph clustering methods [11], [12] have been proposed. These methods typically construct adjacency matrices based on the nearest neighbor graph [13] or attributed graph [14] and learn graph embedding using a graph autoencoder (GAE) [15]. Finally, conventional clustering algorithms, such as K-means [16], are applied based on the learned graph embedding. The GNN-based clustering methods promise resulting promising by learning the node representations from the graph structure and node attributes simultaneously. However, there are still some challenges in GNN-based clustering. Inevitably, there will be some spurious edges in the initial graph, i.e., edges that are not conducive to clustering. These spurious edges make the graph convolutional layers pass information between the nodes that should not belong to the same cluster, deteriorating the clustering performance. However, few GNN-based methods try to propose effective methods for this problem. To mitigate the performance impact of spurious edges in graph clustering, we introduce a pioneering use of graph pooling to select informative nodes and edges, and then use the selected edges and the node representations to train a graph classifier to obtain the final clustering assignments.

In recent years, graph pooling [17] emerged to learn abstract representation of the input graph by summarizing local components and discarding redundant information. With growing interest in graph pooling, some improved methods have been proposed. The current graph pooling methods can be divided into two classes: 1) differentiable graph pooling methods [2], [18], [19], [20] and 2) sorting-based graph pooling methods [21], [22], [23], [24].

Ying et al. [2] first proposed the differentiable graph pooling. It can learn an assignment matrix at each GNN layer and generate hierarchical representation of graphs. To effectively capture the graph substructure, adaptive structure aware pooling (ASAP) [25] proposes a sparse pooling operator capable of capturing local subgraph information hierarchically. Chen et al. [26] designed a pooling module that can capture the spatial layouts of points to learn the hierarchical features adequately and applied it to point cloud segmentation. Differentiable pooling works well in several tasks, but suffers from two main disadvantages in node clustering: 1) there is no clear mapping between the nodes of the previous and subsequent layers, which is crucial for clustering tasks and 2) these methods have a quadratic storage complexity, and the number of its parameters is dependent on the number of nodes.

Sorting-based graph pooling learns a projection vector that is applied to each node feature to obtain a score. Graph U-Net [21] uses a top-k choice of nodes for their gPool layer capable of learning a node score and dropping low score nodes. Cangea et al. [27] applied this to graph classification and proposed TopKPool, which achieved comparable performance with DiffPool. Self-attention graph (SAGPool) [28] takes graph topology into account and devises a SAGPool pooling method for GNNs in the context of hierarchical graph pooling. To preserve diverse representative nodes in different neighborhoods, Gao et al. [29] leveraged the proposed neighborhood information gain criterion to select informative nodes in each neighborhood. Sorting-based methods do not break the mapping from the nodes of the previous layer to the next layer. But, the selected nodes...
may not be distributed across the essential areas in the graph, which may cause absence clusters in clustering tasks.

The methods mentioned earlier generally utilize localized node features and connecting relationships to obtain a smaller graph representation. However, all these graph pooling methods focus on graph representation rather than node representation. Therefore, they can be used to select the nodes and edges that best characterize the graph. But, it is difficult to apply them directly to remove spurious edges in graph clustering.

In this work, to achieve more efficient and accurate nodes clustering on graph, we design a new dual graph embedding network (DGEN), which learns global graph embedding and local graph embedding step by step, with the two processes connected by a graph pooling layer. To reduce the impact of spurious edges on graph clustering, we design a novel neighbor cluster pooling (NCPool), which selects the most informative subset of nodes and the corresponding edges based on the distance of nodes and their nearest neighbors.

The fundamental idea of the pooling is that the nodes and their nearest neighbors should be close to the same clustering center. To obtain the nearest neighbor of each node in the nodes and their nearest neighbors to cluster centers. To select nodes that fit the clustering assignment of the selected nodes, finally, the selected nodes and their labels are used to train a classifier to obtain the final clustering assignments.

Our major contributions can be summarized as follows.

1) We propose a DGEN for clustering on the graph-structured data. To the best of our knowledge, it is the first time to employ graph pooling to select informative nodes for graph clustering.
2) A novel neighbor clustering pooling (NCPool) is devised to capture local graph information, which can effectively enhance the robustness of graph clustering for the spurious edges.
3) Extensive experiments on five benchmark graph datasets show that our DGEN outperforms the state-of-the-art graph clustering methods.

II. PROPOSED METHODOLOGY

To improve the robustness of graph clustering for the spurious edges, we propose a sorting-based graph pooling called NCPool,

which uses node features and graph topology to choose informative nodes and corresponding edges that are more suitable for clustering. Moreover, a novel DGEN based on this layer is proposed, which is shown in Fig. 1. First, a GAE is applied to learn the global graph embedding on which the NCPool is performed. Local graph embedding is then learned based on the selected nodes and edges. Finally, K-means is employed for the selected node representations to obtain the final clustering assignments, and the local clustering assignments are used as labels to train a classifier that can obtain the final clustering assignments. In the following, we will describe our proposed model in detail.

Let a graph be represented by a triple \( G = (V, E, X) \) with \( N = |V| \) nodes and \( |E| \) edges. The \( d \)-dimensional feature of each node \( v_i \in V \) is denoted by \( x_i \), and \( X \in \mathbb{R}^{N\times d} \) denotes the node feature matrix. The topological structure of \( G \) can be represented by an adjacency matrix \( A \), where \( A_{i,j} = 1 \) if \((v_i, v_j) \in E\); otherwise \( A_{i,j} = 0 \).

A. NCPool

In graph embedding, spurious edges can reduce the discriminability of the learned node representation, which, in turn, affects the performance of graph clustering. To reduce the impact of spurious edges on graph clustering, we propose a novel NCPool to select the nodes that are close to the same clustering center as their neighbors.

Here, a pooling function \( S(\cdot) \) is introduced to select the subset \( \Phi \subset V \) containing \( pN \) nodes. Also, the pooling problem can be formulated as follows:

\[
\min_{\Phi \subseteq V} S(\Phi), \quad \text{s.t.} |\Phi| = pN
\]

where \( p \in (0, 1) \) denotes the ratio of selected nodes, and \( S(\Phi) \) is designed based on the distance of the nodes and their nearest neighbors to cluster centers. To select nodes that fit the clustering assignment better, we calculate the cluster centers \( c \) by \( K \)-means [16]. There are two main advantages of using \( K \)-means: 1) it is fast and efficient, with little impact on the speed of model training and 2) in general, graph embedding obeys the Gaussian distribution, which satisfies the assumption of \( K \)-means.

The distance between the node and each cluster center is first calculated, and we can obtain the distance between the node and
the closest cluster center
\[
\sum_{v_i \in \Phi} \| H_{v_i} - H_{c_{ij}} \|_2^2
\]  
(2)
where \( H_{v_i} \) is the node representation of \( v_i \) fed into the pooling layer. Also, \( c_{ij} \) is the closest cluster center to \( v_i \).

To find the nearest neighbor of each node on the attributed graph, we use a modified SNN [30] to define the similarity of the nodes on the graph. Also, the modified SNN similarity is written as follows:
\[
sim(i, j) = \begin{cases} 
0, & (v_i, v_j) \notin \mathcal{E} \\
|\mathcal{N}(i) \cap \mathcal{N}(j)|, & (v_i, v_j) \in \mathcal{E}
\end{cases}
\]  
(3)
where \( \mathcal{N}(s) \) denotes the neighboring nodes of \( v_i \). After getting the similarity of all other points to \( v_i \), the most similar node is selected as the nearest neighbor of \( v_i \).

Then, the final \( S(\Phi) \) can be calculated by the following equation:
\[
S(\Phi) = \sum_{v_i \in \Phi} (\| H_{v_i} - H_{c_{ij}} \|_2^2 + \| H_{v_{in}} - H_{c_{ij}} \|_2^2)
\]  
(4)
where \( v_{in} \) denotes the nearest neighbor of \( v_i \).

After obtaining the score of all nodes, we apply the node selection method proposed by Gao and Ji [21] and Cangea et al., which retains a portion of nodes of the input graph even when graphs of varying sizes and structures are inputted. The top \( kN \) nodes are selected based on the value of \( S \):
\[
\text{idx} = \text{top}_{kN}(S, [kN]).
\]  
(5)
Also, an input graph is processed by the following operation:
\[
H' = H_{\text{idx}} \odot (a^T I), \quad A' = A_{\text{idx, idx}}
\]  
(6)
where \( a \) denotes an affinity vector consisting of the score for the selected nodes, and \( I \) is an all-one vector. \( H_{\text{idx}} \) is the row-wise indexed feature matrix, and \( A_{\text{idx, idx}} \) is the row-wise and col-wise indexed adjacency matrix.

### B. Dual Graph Embedding Network

In this section, the proposed DGEN is described in detail. First, a global graph attention layer (GAT) encoder learns the representation of all nodes using the whole graph structure and all node attributes, which we call the global graph embedding \( H^G \). The initial representation learned by the global GAT encoder makes it easier for the NCPool to select nodes and edges compared with the original data. Then, \( H^G \) is fed into the NCPool to obtain informative nodes and edges, and the final local graph embedding \( H^L \) is obtained by the local GAT encoder with the selected edges and nodes as inputs. The local GAT encoder can continue to learn node representations on the local graph structure to obtain a more discriminative representation than the direct output of selected nodes and edges. Finally, the local clustering process is conducted on the local graph embedding \( H^L \), and the labels obtained are used to train a GAT classifier, which is used to obtain the final clustering assignments for all the nodes.

In our DGEN, the GAT [31] with multthead is used to learn the representation of neighboring nodes adaptively. Let \( h_i \) be the latent representation of the node \( v_i \) learned by the \( l \)th layer, and the latent representation \( h'_i \) learned by the next layer can be expressed as follows:
\[
h'_i = \frac{M}{m=1} \left( \phi \left( \sum_{l=1}^{m} a_{ij}^m W^m h_j \right) \right)
\]  
(7)
where \( \mathcal{N}(i) \) denotes the neighboring nodes of \( v_i \), \( \odot \) means the concatenation operation, and \( W^m \) is a weight matrix that performs a shared linear transformation on \( h \). The attention coefficients \( a_{ij}^m \) can be computed as follows:
\[
a_{ij}^m = \frac{\exp(\text{LeakyReLU}(a^T (W^m h_i) \| W^m h_j))}{\sum_{k \in \mathcal{N}(j)} \exp(\text{LeakyReLU}(a^T (W^m h_i) \| W^m h_k))}
\]  
(8)
where \( a^T \) is a weight vector, and \( \| y \) is the concatenation operation between \( x \) and \( y \).

Each GAT encoder includes two GAT layers. Also, to guide the GAT encoders to learn a comprehensive node representation, an inner product decoder is applied to reconstruct the global graph \( \hat{A} \) from the global embedding \( H^G \) and reconstruct the local graph \( \hat{A} \) from the local embedding \( H^L \):
\[
\hat{A}_{ij} = \text{sigmoid}(h_i^G \cdot (h_j^G)^T)
\]  
(9)
\[
\hat{A}_{ij} = \text{sigmoid}(h_i^L \cdot (h_j^L)^T)
\]  
(10)
where \( \hat{A}_{ij} \) and \( \hat{A}_{ij} \) denote the adjacency of node \( v_i \) and node \( v_j \) in the reconstructed graph \( \hat{A} \) and \( \hat{A} \), respectively. Then, we minimize the global reconstruction error \( L_G \) and the local reconstruction error \( L_L \) as follows:
\[
L_G = \sum_i \text{loss}(A_{ij}, \hat{A}_{ij})
\]  
(11)
\[
L_L = \sum_i \text{loss}(A'_{ij}, \hat{A}'_{ij})
\]  
(12)
where \( A'_{ij} \) denotes the adjacency of the selected node \( v_i \) and \( v_j \) in the adjacency matrix \( A' \).

The aforementioned part merely aims to obtain the representation of selected nodes, but cannot guarantee that the obtained representation is cluster-friendly. Inspired by the good properties of “Kullback-Leibler (KL) divergence”-based clustering [11], [32], we develop a self-optimizing training module as a solution to overcome this difficulty. Specifically, for the local graph embedding \( H^L \), if its cluster center is denoted by \( \mu_j \), the loss function can be written as follows:
\[
L_c = \text{KL}(P \| Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}
\]  
(13)
In (12), \( q_{ij} \) can be calculated using the Student’s t-distribution [33] as follows:
\[
q_{ij} = \left( \frac{1 + \| h_i^L - \mu_j \|_2^2}{\sum_k (1 + \| h_k^L - \mu_j \|_2^2)} \right)^{-1}
\]  
(14)
where \( h_i^L \) is the \( i \)th row of \( H^L \), and \( \mu_j \) is initialized by \( K \)-means on the pretrained representation. \( q_{ij} \) can be seen as the probability of assigning node \( h_i^L \) to cluster \( j \). \( Q = \{q_{ij}\} \) is the distribution of the assignments of all nodes. \( P \) is the target distribution of \( Q \) and can be calculated as follows:
\[
p_{ij} = \frac{q_{ij}^2 / \sum_k q_{ik}}{\sum_k q_{ik}^2 / \sum_l q_{lk}}
\]  
(15)
By minimizing the KL divergence loss between distributions \( P \) and \( Q \), a more cluster-friendly representation can be produced according to the high confidence predictions. Thus, the total objective function of the DGEN is defined as follows:
\[
L = L_G + \lambda L_c
\]  
(16)
where \( \lambda \) is the hyperparameter balancing these two losses.

Now, we can obtain the clustering assignments of the selected nodes. However, our goal is to obtain the cluster assignment of all nodes. To obtain the global clustering assignments, a GAT classifier is trained using the selected nodes, corresponding edges, and local clustering assignments. Also, the final clustering assignments can be
Algorithm 1 DGEN Algorithm

Input: Graph data: $G$, Number of clusters: $C$, Maximum iterations: $MaxIter$, Target distribution update interval: $T$, hyper-parameter: $\lambda$; 

Output: Clustering results $R$;

begin
1 Construct SNN graph according to Eq.(3);
2 Initialize parameters of encoders and classifier;
3 Pre-train the Global GAT Encoder via Eq.(10);
4 Initialize global cluster center $c$ and local cluster center $\mu$;
5 for $iter \in [0, 1, \ldots, MaxIter]$ do
6 Generate global graph embedding $H^G$;
7 Calculate $(\Phi)$ and select nodes using Eq.(4)-(6);
8 Generate local embedding $H^L$ based on the selected nodes;
9 Calculate distribution $Q$ via Eq.(13);
10 if $iter\%T == 0$ then
11 Update distribution $P$ using Eq.(14);
12 end
13 Calculate total loss function via Eq.(15) and update network parameters;
14 end
15 Calculate local clustering results $R_l$ based on $H^L$;
16 Use the local nodes, corresponding edges and local clustering results $R_l$ to train the GAT classifier;
17 Obtain $R$ using the GAT Classifier;
18 return clustering assignments $R$;
end

TABLE I
BENCHMARK GRAPH DATASETS

| Dataset   | Clusters | Nodes | Features | Links |
|-----------|----------|-------|----------|-------|
| Cora$^1$  | 7        | 2708  | 1433     | 5429  |
| CiteSeer$^1$ | 6       | 3327  | 3703     | 4732  |
| PubMed$^1$ | 3        | 19717 | 500      | 44338 |
| DBLP$^2$  | 4        | 4058  | 334      | 7056  |
| ACM$^2$   | 3        | 3025  | 3703     | 26256 |

obtained by inputting all nodes and selected edges to the classifier. There is a notable advantage for our model to use the classifier to obtain the final clustering assignments instead of obtaining them directly based on the adjacent relations. Spurious edges between discard nodes and selected nodes can lead to inaccurate classification. The algorithm of the whole model is shown in Algorithm 1.

III. EXPERIMENTS

A. Experimental Settings

1) Datasets: To evaluate the effectiveness of our proposed method, we conduct experiments on five citation networks widely used to assess the attributed graph analysis. These datasets are summarized in Table I.

2) Evaluation Metrics: Three standard evaluation metrics are as follows: 1) accuracy (ACC); 2) normalized mutual information (NMI); and 3) adjusted rand index (ARI), which are used to evaluate the performance. In all cases, the values of these metrics range between 0 and 1, where higher values correspond to better clustering performance.

3) Baseline Methods: To verify the effectiveness of the proposed DGEN, we compare it with a total of 12 clustering algorithms, including methods that use only node attribute or graph structure, and both. All methods are performed with a known number of clusters.

1) Using Node Attribute: K-means [16] is one of the most commonly used clustering methods.

2) Using Graph Structure: Spectral clustering [34], denoising autoencoder for graph embedding (DNGR) [35], and modularized nonnegative matrix factorization (M-NMF) [36] cluster nodes by their connectivity relationships in the graph.

3) Using Both Node Attribute and Graph Structure: GAE and variational GAEs (VGAEs) [15], marginalized GAE (MGAE) [37], adversarial regularized GAE (ARGAE) and adversarial variational regularized GAE (ARVGAE) [38], deep attentional embedding graph clustering (DAEGC) [11], graph attention auto-encoders (GATE) [31], and distribution-induced bidirectional generative adversarial network (DBGAN) [39] obtain clustering results by learning the node representations from the initial graph structure and node attributes.

4) Implementation Details: In our experiments, the ratio of selected nodes of NCPool is set to 0.6, and the hyperparameter $\lambda$ is set to 10. Moreover, there are three main network modules in our DGEN. The detailed configurations of these network modules are given in Table II. Note that the local GAT encoder is constructed with a ten-neuron embedding layer for all datasets, which is much smaller than the commonly used 64-neuron. Adam optimizer is used with a learning rate of $\alpha = 0.001$ for the GAT encoders and $\alpha = 0.0005$ for the GAT classifier.

B. Comparison of Clustering Performance

1) Evaluation Metrics: The results on the five benchmark datasets are reported in Table III, where the top value is highlighted in red font and the second best in blue. $X$, $G$, and $G\&X$ indicate that the method uses only node attribute, graph structure, or both attribute and structure information, respectively. According to these results, we have the following observations. First, our proposed method DGEN outperforms the competing methods on these benchmark datasets in most cases. For example, the DGEN shows a significant improvement over the base method GATE in ACC and ARI on Cora by a margin of around 11.3% and 11.5%, respectively. Second, the methods using both node attribute and graph structure consistently obtain better performances than methods using only one type of information, which demonstrates that both the graph structure and the node attribute contain useful information for graph clustering. Third, the DGEN achieves the competitive results than other GNN-based methods, which indicates the feasibility of applying graph pooling to graph clustering and the effectiveness of our proposed NCPool. The second best baseline (DBGAN) indeed outperforms our DGEN in several cases. This, however, is achieved at the cost of a higher representation dimension.

2) Visualization: To show the superiority of the representation obtained by our method, we visualize the results of different methods on Cora and PubMed with t-distributed stochastic neighbor
TABLE III

| Method   | Info. | ACC   | NMI   | ARI   | ACC   | NMI   | ARI   | ACC   | NMI   | ARI   |
|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| K-means  | X     | 0.500 | 0.317 | 0.239 | 0.544 | 0.312 | 0.285 | 0.580 | 0.278 | 0.246 |
| Spectral | G     | 0.398 | 0.297 | 0.174 | 0.308 | 0.090 | 0.082 | 0.496 | 0.147 | 0.098 |
| DNGR     | G     | 0.419 | 0.318 | 0.142 | 0.326 | 0.180 | 0.043 | 0.468 | 0.153 | 0.059 |
| M-NMF    | G     | 0.423 | 0.256 | 0.161 | 0.336 | 0.099 | 0.070 | 0.470 | 0.084 | 0.058 |
| GAE      | G&E  | 0.530 | 0.397 | 0.293 | 0.380 | 0.174 | 0.141 | 0.632 | 0.249 | 0.246 |
| VGEA     | G&X  | 0.592 | 0.408 | 0.347 | 0.392 | 0.163 | 0.101 | 0.619 | 0.216 | 0.201 |
| MGAE     | G&X  | 0.684 | 0.511 | 0.448 | 0.661 | 0.412 | 0.414 | 0.593 | 0.282 | 0.248 |
| ARGAE    | G&X  | 0.640 | 0.449 | 0.352 | 0.573 | 0.350 | 0.341 | 0.681 | 0.276 | 0.291 |
| ARVGA    | G&X  | 0.638 | 0.450 | 0.374 | 0.544 | 0.261 | 0.245 | 0.513 | 0.117 | 0.078 |
| DAGEC    | G&X  | 0.704 | 0.528 | 0.496 | 0.672 | 0.397 | 0.410 | 0.671 | 0.266 | 0.278 |
| GATE     | G&X  | 0.658 | 0.527 | 0.451 | 0.616 | 0.401 | 0.381 | 0.673 | 0.323 | 0.299 |
| DBGAN    | G&X  | 0.748 | 0.560 | 0.540 | 0.670 | 0.407 | 0.414 | 0.694 | 0.324 | 0.327 |
| DGEN     | G&X  | 0.771 | 0.576 | 0.566 | 0.686 | 0.430 | 0.445 | 0.695 | 0.297 | 0.320 |
| CiteSeer | ACC   | 0.688 | 0.430 | 0.445 | 0.695 | 0.297 | 0.320 | 0.667 | 0.340 | 0.306 |
| PubMed   | ACC   | 0.667 | 0.340 | 0.306 | 0.890 | 0.629 | 0.708 | 0.884 | 0.653 | 0.699 |

Fig. 2. t-SNE visualizations of representations learned by various methods on Cora (top row) and PubMed (bottom row). (a) Cora(raw). (b) Cora(GAE). (c) Cora(GATE). (d) Cora(DGEN-G). (e) Cora(DGEN-L). (f) PubMed(raw). (g) PubMed(GAE). (h) PubMed(GATE). (i) PubMed(DGEN-G). (j) PubMed(DGEN-L).

TABLE IV

| Method    | Cora ACC | NMI | ARI | CiteSeer ACC | NMI | ARI | PubMed ACC | NMI | ARI |
|-----------|----------|-----|-----|-------------|-----|-----|-----------|-----|-----|
| TopKPool  | 0.682    | 0.296 | 0.178 | 0.571        | 0.354 | 0.310 | 0.544        | 0.165 | 0.166 |
| SAGPool   | 0.654    | 0.525 | 0.478 | 0.602        | 0.335 | 0.335 | 0.593        | 0.196 | 0.136 |
| NCPool    | 0.771    | 0.576 | 0.566 | 0.686        | 0.430 | 0.445 | 0.695        | 0.297 | 0.320 |

Fig. 3. Clustering results with different λ values. (a) ACC. (b) NMI.

C. Parameter Sensitivity Analysis

1) Impact of Hyperparameter λ: In our proposed model, there is a main hyperparameter λ that balances the reconstruction loss and the clustering loss. In the following, we will analyze the sensitivity of this parameter in the proposed method. In the experiments, we tune λ from {10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4}, and the ACC and NMI of the final clustering results with different λ values are shown in Fig. 3. As shown in Fig. 3, when λ is between 10 and 100, the corresponding evaluation metrics ACC and NMI will largely remain constant.

2) Impact of Ratio of Reserved Nodes: The current graph pooling methods usually set the ratio to 0.5. Since our goal is to learn the representations of nodes rather than the representation of the graph, we do not use the common settings in our approach. We search it in the range of [0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]. Also, Fig. 4 shows the results of our method using different ratios of reserved nodes (taking CiteSeer as an example). It is observed that the promising performance could be expected when the ratio of reserved nodes is...
between 0.6 and 0.7. The reason we cannot choose a smaller ratio as other graph pooling methods is that dropping too many nodes can lead to larger errors when classifying all nodes. Hence, the ratio of selected nodes of NCPool is set to 0.6 for all datasets.

### D. Model Analysis

1) **Effectiveness of Spurious Edges Removal:** As mentioned before, spurious edges can weaken the discriminability of the learned node representations. To validate this, we compare the clustering results of the local node representations learned with or without spurious connections. Specifically, we compare the clustering results of the selected node representations (DGEN-s) and the representation of the corresponding nodes in the global graph embedding (DGEN-g).

![Figure 4](image.png)

**Figure 4.** Parameter effect of the ratio of reserved nodes.

| Method   | ACC | Cora NMI | ARI |
|----------|-----|----------|-----|
| DGEN     | 0.80 | 0.50 | 0.60 |
| DGEN-N    | 0.78 | 0.48 | 0.56 |
| DGEN-L    | 0.77 | 0.57 | 0.56 |
| DGEN      | 0.77 | 0.57 | 0.56 |

### TABLE V

| Method   | ACC | Cora NMI | ARI |
|----------|-----|----------|-----|
| DGEN     | 0.80 | 0.50 | 0.60 |
| DGEN-N    | 0.78 | 0.48 | 0.56 |
| DGEN-L    | 0.77 | 0.57 | 0.56 |
| DGEN      | 0.77 | 0.57 | 0.56 |

### TABLE VI

| Method   | ACC | Cora NMI | ARI |
|----------|-----|----------|-----|
| DGEN     | 0.80 | 0.50 | 0.60 |
| DGEN-N    | 0.78 | 0.48 | 0.56 |
| DGEN-L    | 0.77 | 0.57 | 0.56 |
| DGEN      | 0.77 | 0.57 | 0.56 |

### TABLE VII

| Method   | ACC | Cora NMI | ARI |
|----------|-----|----------|-----|
| DGEN     | 0.80 | 0.50 | 0.60 |
| DGEN-N    | 0.78 | 0.48 | 0.56 |
| DGEN-L    | 0.77 | 0.57 | 0.56 |
| DGEN      | 0.77 | 0.57 | 0.56 |

### TABLE VIII

| Method   | ACC | Cora NMI | ARI |
|----------|-----|----------|-----|
| DGEN     | 0.80 | 0.50 | 0.60 |
| DGEN-N    | 0.78 | 0.48 | 0.56 |
| DGEN-L    | 0.77 | 0.57 | 0.56 |
| DGEN      | 0.77 | 0.57 | 0.56 |

Obviously, it is more efficient to use the classification step to obtain the final clustering results, because obtaining the final clustering results based on adjacency requires spurious edge information, which can undermine the clustering performance.

3) **NCPool With Different Clustering Center Acquisitions:** In our model, K-means is used to find the cluster centers, and it can be replaced by other clustering methods. To demonstrate that the NCPool is insensitive to the clustering center acquisition, we use Gaussian mixture models (GMMs) in place of K-means, and the results are reported in Table IX. The NCPool-GMM in Table VII shows the best performance compared with diverse baselines. The NCPool-GMM outperforms the NCPool-KM in most cases. It proves that our model is not sensitive to clustering center acquisition.

4) **Effectiveness of the Structure of DGEN:** DGEN mainly consists of the global GAT encoder, the NCPool, the local GAT encoder, and a graph classifier. We then conduct an experiment to verify the necessity of the global GAT encoder and the local GAT encoder. The clustering results are shown in Table IX. The DGEN-GN denotes that it consists of the global GAT encoder and the NCPool, and the DGEN-NL denotes that it consists of the NCPool and the local GAT encoder. Obviously, the whole structure of the DGEN provides higher clustering performance, which demonstrates the effectiveness of the global GAT encoder and the local GAT encoder.

### IV. Conclusion

In this brief, we propose a DGEN to learn a more robust representation for graph clustering. Similar to most of the graph learning methods, we first learn the representation of all nodes. To reduce the effect of spurious edges on graph clustering, we design a novel NCPool, which can select an informative subset of nodes and the corresponding edges. The pooling is based on the assumption that if a node and its nearest neighboring node are close to the same clustering center, this node is an informative node, and the corresponding edge can be considered as a cluster-friendly edge. Then, the local graph embedding is learned based on the selected nodes and edges. To train the whole model, we apply self-optimizing clustering loss and reconstruction loss to make the learned representation more suitable for clustering. Consequently, the conventional clustering algorithm K-means is performed on the selected node representations to obtain the local clustering assignments. Finally, we use the local clustering assignments as labels to train a classifier that enables us to obtain the clustering assignments of all nodes. An evaluation on several benchmark datasets demonstrates the effectiveness of DGEN compared with diverse baselines. In the future, we plan to explore graph node clustering and graph clustering with an unknown number of clusters, as the number of clusters is hard to obtain in some applications. Moreover, we are going to further improve the performance of node selection by graph contrastive learning and apply graph pooling to more fields.

### REFERENCES

[1] X. Zhang, C. Xu, X. Tian, and D. Tao, “Graph edge convolutional neural networks for skeleton-based action recognition,” *IEEE Trans. neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 3047–3060, Aug. 2019.

[2] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 4805–4815.

[3] S. Dhelin, N. Aung, and H. Ning, “Mining user interest based on personality-aware hybrid filtering in social networks,” *Knowl.-Based Syst.*, vol. 206, Oct. 2020, Art. no. 106227.

[4] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[21] H. Gao, Y. Liu, and S. Ji, “Topology-aware graph pooling networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 12, pp. 4512–4518, Dec. 2021.
[22] Y. Hu, J. Gao, and C. Xu, “Learning dual-pooling graph neural networks for few-shot video classification,” IEEE Trans. Multimedia, vol. 23, pp. 4285–4296, 2021.
[23] E. Ranjan, S. Sanyal, and P. P. Talukdar, “ASAP: Adaptive structure aware pooling for learning hierarchical graph representations,” in Proc. AAAI Conf. Artif. Intell. (AAAI), 2020, pp. 5470–5477.
[24] C. Chen, S. Qian, Q. Fang, and C. Xu, “HAPGN: Hierarchical attentive pooling graph network for point cloud segmentation,” IEEE Trans. Multimedia, vol. 23, pp. 2335–2346, 2021.
[25] C. Cangea, P. Velicković, N. Jovanović, T. Kipf, and P. Liò, “Towards sparse hierarchical graph classifiers,” 2018, arXiv:1811.01287.
[26] J. Lee, I. Lee, and J. Kang, “Self-attention graph pooling,” in Proc. 39th Int. Conf. Mach. Learn. (ICML), 2019, pp. 3734–3743.
[27] X. Gao, W. Dai, C. Li, H. Xiong, and P. Frossard, “iPool—Information-based pooling in hierarchical graph neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 9, pp. 5032–5044, Sep. 2022.
[28] R. A. Jarvis and E. A. Patrick, “Clustering using a similarity measure based on shared near neighbors,” IEEE Trans. Comput., vol. C-100, no. 11, pp. 1025–1034, Nov. 1973.
[29] A. Salehie and H. Davulcu, “Graph attention auto-encoders,” in Proc. IEEE 32nd Int. Conf. Tools Artif. Intell. (ICTAI), Nov. 2020, pp. 989–996.
[30] J. Xie, R. B. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in Proc. 33rd Int. Conf. Mach. Learn. (ICML), 2016, pp. 478–487.
[31] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, pp. 2579–2605, Nov. 2008.
[32] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2001, pp. 849–856.
[33] S. Cao, W. Lu, and Q. Xu, “Deep neural networks for learning graph representations,” in Proc. AAAI Conf. Artif. Intell. (AAAI), 2016, pp. 1145–1152.
[34] W. Xiao, C. Peng, W. Jing, P. Jian, Z. Wenwu, and Y. Shiqiang, “Community preserving network embedding,” in Proc. AAAI Conf. Artif. Intell. (AAAI), 2017, pp. 203–209.
[35] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, “MGAE: Marginalized graph autoencoder for graph clustering,” in Proc. ACM Conf. Inf. Knowl. Manag. (ACM CIKM), 2017, pp. 889–898.
[36] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, “Adversarially regularized graph autoencoder for graph embedding,” in Proc. Int. Joint Conf. Artif. Intell. (IJCAI), 2018, pp. 2699–2705.
[37] S. Zheng, Z. Zhu, X. Zhang, Z. Liu, J. Cheng, and Y. Zhao, “Distribution-induced bidirectional generative adversarial network for graph representation learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 7222–7231.