



Full length article

Deep cut-informed graph embedding and clustering

Zhiyuan Ning^{a,b,c},¹ Zaitian Wang^{a,b},¹ Ran Zhang^{a,b}, Ping Xu^{a,b}, Kunpeng Liu^d,
 Pengyang Wang^e, Wei Ju^f, Pengfei Wang^{a,b},^{*} Yuanchun Zhou^{a,b}, Erik Cambria^g,
 Chong Chen^h

^a Computer Network Information Center, Chinese Academy of Sciences, China

^b University of Chinese Academy of Sciences, China

^c School of Engineering, Westlake University, China

^d Department of Computer Science, Portland State University, United States of America

^e Department of Computer and Information Science, The State Key Laboratory of Internet of Things for Smart City, University of Macau, China

^f College of Computer Science, Sichuan University, China

^g College of Computing and Data Science, Nanyang Technological University, Singapore

^h Future City Lab, Terminus Group, China

ARTICLE INFO

Keywords:

Graph embedding

Deep clustering

Cut-informed

ABSTRACT

Graph clustering aims to divide the graph into different clusters. The recently emerging deep graph clustering approaches are largely built on graph neural networks (GNN). However, GNN is designed for general graph encoding and there is a common issue of representation collapse in existing GNN-based deep graph clustering algorithms. We attribute two main reasons for such issues: (i) the inductive bias of GNN models: GNNs tend to generate similar representations for proximal nodes. Since graphs often contain a non-negligible amount of inter-cluster links, the bias results in error message passing and leads to biased clustering; (ii) the clustering guided loss function: most traditional approaches strive to make all samples closer to pre-learned cluster centers, which causes a degenerate solution assigning all data points to a single label thus making all samples similar and less discriminative. To address these challenges, we investigate graph clustering from a graph cut perspective and propose an innovative and non-GNN-based Deep Cut-informed Graph embedding and Clustering framework, namely *DCGC*. This framework includes two modules: (i) cut-informed graph encoding; (ii) self-supervised graph clustering via optimal transport. For the encoding module, we derive a cut-informed graph embedding objective to fuse graph structure and attributes by minimizing their joint normalized cut. For the clustering module, we utilize the optimal transport theory to obtain the clustering assignments, which can balance the guidance of “proximity to the pre-learned cluster center”. With the above two tailored designs, DCGC is more suitable for the graph clustering task, which can effectively alleviate the problem of representation collapse and achieve better performance. We conduct extensive experiments to demonstrate that our method is simple but effective compared with benchmarks.

1. Introduction

Graph clustering is a fundamental task that has attracted intensive attention and achieved great success on many fields, such as co-saliency and community detection [1–3]. Beside graph structure, each sample/node is usually accompanied with attributes, such as keywords in citation networks and covariates in social networks. Many researchers have paid attention to combining the graph structure and node attributes to elevate graph clustering performances.

Recently, with the emergence of deep learning [4,5], deep graph clustering has become a rapidly evolving research area, which addresses the task of training neural networks to learn representations and partitioning nodes in attribute graphs into distinct groups. Promising improvements have been made compared with conventional methods in such an unsupervised mission through utilizing deep neural networks, particularly graph neural networks (GNNs) [6–8]. These models typically embed nodes into the hidden space and then perform clustering algorithms. In particular, the work in [9] jointly considered the local structure preservation in deep clustering, optimizing cluster

* Correspondence to: CAS Informatization Plaza No.2 Dong Sheng Nan Lu, Haidian District, Beijing 100083, China

E-mail address: pfwang@cnic.cn (P. Wang).

¹ Equal contribution.

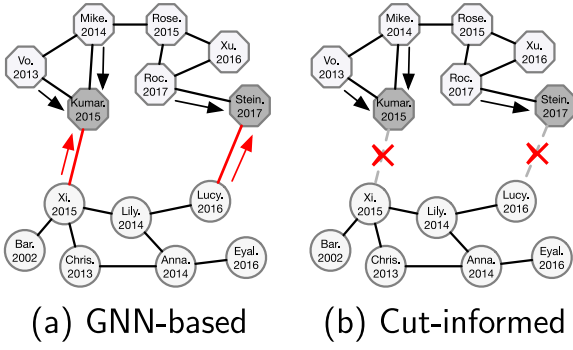


Fig. 1. The motivation of our deep cut-informed graph embedding and clustering: (a) the GNN-based methods can lead to error message passing when noisy links exist. (b) Our cut-informed approach learns embeddings corresponding to minimal normalized cut.

assignments and learning features by integrating the clustering loss and auto-encoder reconstruction loss. The work in [10] designed a delivery operator and a dual self-supervised mechanism to combine the auto-encoder representation and the graph convolutional networks (GCNs) [6] representation. The work in [11] proposed a dynamic cross-modality fusion mechanism and a triplet self-supervised strategy, etc. They all focus on developing sophisticated graph neural networks to fuse graph structure and node attributes to learn a general representation of graph topology (e.g., global, local, neighbor, subgraph) and attributes.

Despite achieving commendable performance, there is a common issue in existing GNN-based deep graph clustering algorithms. Specifically, these algorithms often encounter the problem of representation collapse [12], wherein nodes belonging to distinct categories are frequently mapped to similar representations during the sample encoding process. Consequently, the resulting node representations lack discrimination, leading to limited efficacy in clustering performance. The issue of representation collapse remains unresolved, which is restricting the performance of deep graph clustering algorithms that rely on GNNs [12,13].

We attribute two main reasons for the representation collapse problem of GNN-based deep graph clustering algorithms: (i) **The inductive bias of GNNs**: Many existing GNNs have the inductive bias that linked nodes typically belong to the same class or have similar characteristics [14]. Such inductive bias makes GNN representations similar for nodes in close proximity [15]. When the GNNs get deeper, the inductive bias will even lead to the over-smoothing issue (indistinguishable representations of nodes in different classes) [16]. So when faced with graph clustering, an unsupervised learning task without any labeling information, the problem caused by the inductive bias of GNN models will become more severe. Besides, the graph structure usually contains a non-negligible amount of inter-cluster links. For example, citation networks usually have cross-subject links (e.g., Fig. 1(a)). According to the inductive bias, the GNN-based clustering methods may make the node embeddings from different clusters similar, because there may exist error message passing [17] through the inter-cluster links (i.e., red lines in Fig. 1(a)). Thus, it is urgent to design a new method to reduce the effects of error propagation caused by GNNs' inductive bias. (ii) **The clustering guided loss function**: Most traditional approaches adopt a clustering guided loss function [18] to force the generated sample embeddings to have the minimum distortion against the pre-learned clustering centers. They typically strive to minimize the distance between data representations and pre-learned cluster centers, thereby enhancing cluster cohesion [10,19–21]. However, making all samples much closer to cluster centers may cause a degenerate solution that assigns all data points to a single (arbitrary) label, thus making all samples similar and less discriminative [12]. Therefore, we need to design a mechanism to balance the guidance of “proximity to the pre-learned cluster center”.

To tackle the above-mentioned issues, we propose a novel and non-GNN-based graph clustering paradigm named Deep Cut-informed Graph embedding and Clustering framework (DCGC). This paradigm includes two modules: (i) **cut-informed graph encoding**, which does not use any GNN-based model as graph encoder and designs the core module from a perspective (i.e., graph cut) that is more reasonable and appropriate for the graph clustering task, therefore avoids the problems caused by the inductive bias of GNN models; (ii) **self-supervised graph clustering via optimal transport**, which balances the guidance of “proximity to the pre-learned cluster center” via optimal transport, thus avoiding the degenerate solutions. Specifically, for the cut-informed graph encoding module, we show that the normalized cut [22] minimization can be relaxed into solving k -smallest eigenvalues of graph Laplacian, where the eigenvectors are continuous approximations of graph partitions and can be viewed as cut-informed graph embeddings. As seen in Fig. 1(b), the cut-informed graph embeddings are much more robust to inter-cluster links. Inspired by covariates-assisted spectral clustering [23], we come up with a cut-informed graph embedding objective function to minimize the joint normalized cut of the original graph (i.e., the structure information of graph) and attribute graph (i.e., the attribute information of nodes). The embeddings learned by DCGC can better capture the latent clustering structure than existing methods. For the self-supervised graph clustering via optimal transport module, we apply an optimal transport [24] based self-supervised strategy to obtain more stable and balanced clustering assignments. Specifically, it transports the original clustering assignments to a new label distribution that is consistent with the cluster size, which can avoid the degenerate solutions of assigning all data points to a single (arbitrary) label. Then we force the clustering assignments to be close to the transported distribution by Kullback–Leibler (KL) divergence loss. With the above two tailored designs, DCGC is more suitable for the task of graph clustering, which can effectively alleviate the problem of representation collapse and achieve better performance.

Our key contributions can be summarized as follows:

- We attribute the problem that the current GNN-based deep graph clustering methods suffer from the phenomenon of representation collapse to two reasons: (i) the inductive bias of GNN models and (ii) the clustering guided loss function. To tackle these issues, we investigate graph clustering from a graph cut perspective and propose a novel and non-GNN-based graph clustering paradigm, DCGC.
- We propose a cut-informed graph encoding module to fuse the graph structure information and the node attribute information by minimizing their joint normalized cut, thus avoiding the use of GNN models and making our paradigm much more appropriate and suitable for the graph clustering task. We show that the normalized cut minimization can be relaxed into solving k -smallest eigenvalues of graph Laplacian, where the eigenvectors are continuous approximations of graph partitions and can be viewed as cut-informed graph embeddings.
- We propose self-supervised graph clustering via optimal transport to balance the guidance of “proximity to the pre-learned cluster center” via optimal transport, thus avoiding the degenerate solution of assigning all data points to a single (arbitrary) label.
- We conduct extensive experiments on 6 challenging real-world graph datasets and the outcomes show our approach can achieve competitive or even superior performance compared with the state-of-the-art deep graph clustering models. We also perform comprehensive ablation experiments to demonstrate that each component of our method is indispensable.

2. Related work

2.1. Attributed graph embedding

Graph encoding transfers the graph node into vectors [25–28]. Attributed graph embedding methods assume node attribute information

Table 1
Notations and descriptions.

Notations	Descriptions
\mathcal{G}, \mathbf{S}	Original graph, and corresponding attribute graph
$\mathcal{V}, \mathbf{X}, \mathcal{E}, \mathbf{A}$	The set of nodes, the attribute features of nodes, the set of edges, and the adjacency matrix
\mathbf{h}_i, \mathbf{H}	The low-dimensional embedding of node v_i , and the set of latent embeddings for all nodes
\mathbf{I}	Identity matrix
\mathbf{D}, \mathbf{D}_s	Degree matrix of original graph, and degree matrix of corresponding attribute graph
L_G, L_S	The normalized graph Laplacian of \mathcal{G} , and the normalized graph Laplacian of \mathbf{S}
Tr	The trace of a matrix
\mathbf{u}_j	Center of cluster j
q_{ij}, Q	The real probability distribution of assigning sample i to cluster j , and that of all samples
p_{ij}, P	The target distribution of sample i , and that of all samples
\hat{P}	The optimal transport plan matrix
π	The estimated proportion of points for each cluster (computed during optimization)
H	The entropy function
λ	The smoothness parameter controlling the equilibrium of clusters in Sinkhorn iteration (hyperparameter)

is available and exploit both topological information and attribute features simultaneously [29–31]. [32] proved that DeepWalk can be interpreted as a factorization approach and proposed an extension method to explore node features. DANE [33] deals with the dynamic environment with an incremental matrix factorization approach, and LANE [34] incorporates the label information into the optimization process to learn a better embedding. Though these methods improve the clustering performance, they fall short in combining the attribute and graph information for graph clustering.

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 [9,18]. Auto-encoder [35] is one of the most commonly used unsupervised deep neural networks, which plays a crucial role in deep clustering. DEC [18] is the most popular method which uses 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 [10,11,19]. DAEGC [19] encodes the topological structure and node contents by introducing the attentional neighbor-wise fusion strategy on the GAE framework. ARGAE [20] adversarially regularized GAE and further improved the clustering performance by introducing an adversarial learning scheme to learn the graph embedding. SDCN [10] designs a delivery operator and a dual self-supervised mechanism. And DFCN [11] designs a dynamic cross-modality fusion mechanism and a triplet self-supervised strategy. MGAI [36] leverages modularity maximization as a contrastive pretext task to effectively uncover the underlying information of communities in graphs. Although these algorithms are well designed for graph data, they are not suitable for unsupervised graph clustering since graph structure often contains a non-negligible amount of inter-cluster links, which lead to considerable error when leveraging structural information by means such as message passing between clusters.

2.3. Optimal transport

Optimal transport was initially introduced as a solution to the problem of minimizing costs during the simultaneous movement of multiple items [24]. In recent years, its application has garnered significant attention from the machine learning and computer vision community,

particularly in the context of comparing distributions represented as feature sets [37]. The remarkable ability of optimal transport to match distributions has led to its utilization in various theoretical and practical tasks, including generative models [38], structural matching [39], and clustering [40]. An acknowledged drawback of optimal transportation lies in its computational complexity, necessitating the exploration of approximation techniques [41]. In the pursuit of an efficient solution, a regularization approach influenced by probabilistic theory has been proposed [42], in conjunction with Sinkhorn’s algorithm. Alternatively, the problem has been relaxed to achieve a quadratic-time solution by eliminating a constraint [43]. Moreover, the incorporation of a kernel method has been introduced as a means to approximate optimal transport [44]. In this paper, we use optimal transport to get a more appropriate target distribution P which has the constraint that the label distribution must be consistent with the mixing proportions, thus avoiding the degenerate solutions by assigning all data points to a single (arbitrary) label.

3. Problem formulation

Given a graph $\mathcal{G} = (\mathcal{V}, \mathbf{X}, \mathcal{E}, \mathbf{A})$, \mathcal{V} is a set of N nodes in the graph, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]^T$ is the attribute features of nodes where $\mathbf{x}_i \in \mathbb{R}^F$ and F is the number of attributes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set of graph and $\mathbf{A} \in \{0, 1\}^{N \times N}$ is the adjacency matrix of graph. For $\forall v_i, v_j \in \mathcal{V}$, $A_{ij} = 1$ if there exists an edge between v_i and v_j , otherwise, $A_{ij} = 0$. Given a vector \mathbf{u} , we write $\|\mathbf{u}\|_2$ as the its Euclidean norm. Given a subset $U \subseteq \mathcal{V}$, we write $|U|$ as the number of nodes in U . The graph clustering task aims to map each node $v_i \in \mathcal{V}$ to the low-dimensional embedding $\mathbf{h}_i \in \mathbb{R}^d$ based on its original attributes $\mathbf{x}_i \in \mathbb{R}^F$ and the graph structure, and separates the node set \mathcal{V} into K disjoint subsets $\mathcal{V} = \mathcal{V}_1 \cup \dots \cup \mathcal{V}_K$ such that each \mathcal{V}_k is corresponding to a specific semantic. Assuming $\mathbf{H}_{N \times d} = [\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_N]^T$ is the latent embedding and $\mathbf{c} = [c_1, \dots, c_N]$ is the clustering assignment, we will simultaneously learn the embedding \mathbf{H} and clustering assignments \mathbf{c} (see Table 1).

4. Method

In this section we introduce the Deep Cut-informed Graph embedding and Clustering framework (DCGC). Specifically, we propose (i) cut-informed graph encoding, which does not use any GNN-based model as graph encoder and design the core module from a perspective (i.e., graph cut) that is more reasonable and appropriate for the graph clustering task, therefore avoids the problems caused by the inductive bias of GNN models; (ii) self-supervised graph clustering via optimal transport, which balances the guidance of “proximity to the pre-learned cluster center” via optimal transport, thus avoids the degenerated solutions (see Fig. 2).

4.1. Cut-informed graph encoding

To avoid the problems caused by the inductive bias of GNNs that they tend to generate similar representations for nodes in close proximity, instead of using any GNN models as graph encoder, we simply use an MLP to encode the inputted attributes, which can be formulated as:

$$\mathbf{H} = \text{MLP}(\mathbf{X}) \quad (1)$$

For graph \mathcal{G} , denote $\mathbf{D} = \text{diag}([d_1, \dots, d_N])$, where d_i is the degree of node i . The normalized graph Laplacian of \mathcal{G} can be formulated as:

$$L_G = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}. \quad (2)$$

Given two subsets of \mathcal{V} denoted as \mathcal{V}_1 and \mathcal{V}_2 , we define $\mathbf{A}(\mathcal{V}_1, \mathcal{V}_2) = \sum_{i \in \mathcal{V}_1, j \in \mathcal{V}_2} A_{ij}$. Then for a partition $\mathcal{V} = \mathcal{V}_1 \cup \dots \cup \mathcal{V}_K$, its normalized cut [22] can be defined as:

$$\text{Ncut}(\mathcal{V}_1, \dots, \mathcal{V}_K) = \frac{1}{2} \sum_{i=1}^K \frac{\mathbf{A}(\mathcal{V}_i, \bar{\mathcal{V}}_i)}{\text{vol}(\mathcal{V}_i)} \quad (3)$$

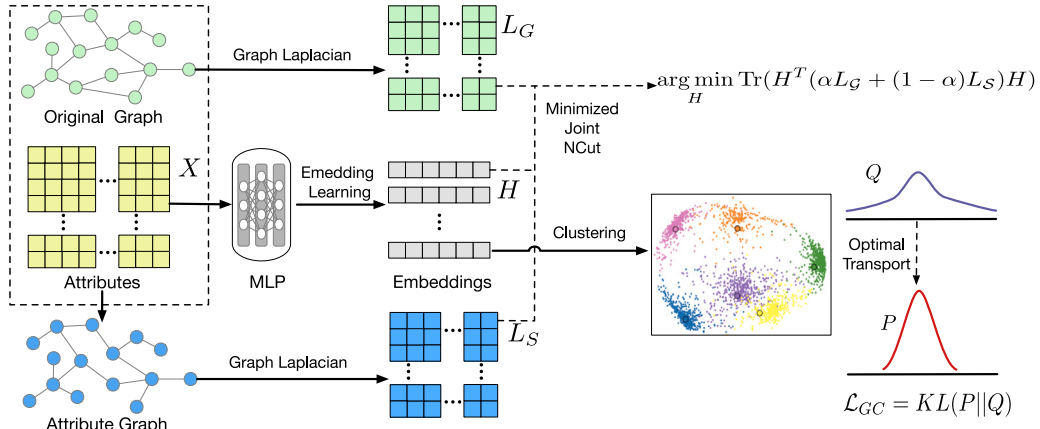


Fig. 2. Framework Overview of Cut-informed Graph Clustering: Given original graph and attributes, attribute graph is constructed. Then attributes are encoded via an MLP to obtain the embedding by minimizing the joint normalized cut of original and attribute graphs. The clustering assignments are optimized by a self-supervised strategy with optimal transport target.

where $\bar{\mathcal{V}}_i$ is the complementary set of \mathcal{V}_i and $\text{vol}(\mathcal{V}_i) = \mathbf{A}(\mathcal{V}_i, \mathcal{V}_i) + \mathbf{A}(\mathcal{V}_i, \bar{\mathcal{V}}_i)$ is the total edges of \mathcal{V}_i . According to this criterion, an optimal graph partition should have minimized Ncut. However, it is NP hard to find the optimal graph partition. Inspired by covariates-assisted spectral clustering [23], we can alternatively solve it with the following optimization strategy:

$$\begin{aligned} & \arg \min_H \text{Tr}(H^T L_G H) \\ & \text{subject to } H^T H = I, \end{aligned} \quad (4)$$

which is a relaxation of minimizing the normalized cut. H is a soft partition of graph \mathcal{G} in terms of minimizing normalized cut, which implies that H can be viewed as the node embeddings for clustering. However, the learned embedding H by Eq. (4) only contains the graph structure information and ignores the node attribute information, so it is not optimal for clustering. Inspired by covariates-assisted spectral clustering [23], we construct an attribute graph S which is defined as $S_{ij} = \left(1 + \frac{x_i^T x_j}{\|x_i\|_2 \|x_j\|_2}\right) / 2$ to model the attribute information. It is easy to see $S_{ij} \in [0, 1]$. We can also define the graph Laplacian of S as:

$$L_S = I - D_S^{-1/2} S D_S^{-1/2}, \quad (5)$$

where D_S is a diagonal matrix that corresponding to the row-sums of S . It is a natural idea to make the embedding H contain the structure information of both original graph (i.e., the structure information of graph) and attribute graph (i.e., the attribute information of nodes), so we have the follow optimization function:

$$\begin{aligned} & \arg \min_H \text{Tr}(H^T(\alpha L_G + (1 - \alpha)L_S)H) \\ & \text{subject to } H^T H = I, \end{aligned} \quad (6)$$

where α is the balance parameter between the original graph and the attribute graph. So the graph encoding loss function can be written as:

$$\mathcal{L}_{GE} = \text{Tr}(H^T(\alpha L_G + (1 - \alpha)L_S)H) + \beta \|H^T H - I\|_F, \quad (7)$$

where β is a tuning parameter and $\|H^T H - I\|_F$ is the loss to guarantee the orthogonality. We chose this weighted sum formulation (Eq. (6)) due to its balance of effectiveness and simplicity observed across diverse datasets during preliminary exploration, allowing for a clear evaluation of our core contributions.

The learned embedding H is suitable for clustering from two viewpoints: (1) The loss function is a relaxation of minimizing the joint normalized cut of the original graph and attribute graph, which means H is a soft assignment of the graph partition from the perspective of minimizing normalized cut. (2) The trade-off between the original graph and attribute graph makes H consistent with both graphs and successfully fuses structure information and attribute information in an elegant and non-GNN-based way.

4.2. Self-supervised graph clustering via optimal transport

To generate clustering assignments, we use a self-supervised strategy proposed by [18]. It uses the Student's t-distribution [45] as a kernel to measure the similarity between embedded point h_i and the clustering center μ_j :

$$q_{ij} = \frac{(1 + \|h_i - \mu_j\|^2 / \theta)^{-\frac{1+\theta}{2}}}{\sum_{j'} (1 + \|h_i - \mu_{j'}\|^2 / \theta)^{-\frac{1+\theta}{2}}} \quad (8)$$

where h_i is encoded embedding of x_i and θ is the degrees of freedom of the Student's t-distribution [45]. q_{ij} can be viewed as the soft assignment of each sample to the clustering centers i.e., the probability of assigning sample i to cluster j . Then, we define $Q = [q_{ij}]$ as the distribution of the assignments of all samples.

After obtaining the clustering result distribution Q , we aim to optimize the data representation by learning from the high confidence assignments. Most traditional approaches adopt a clustering guided loss function [18] to force the generated sample embeddings to have the minimum distortion against the pre-learned clustering centers. They typically strive to minimize the distance between data representations and cluster centers, thereby enhancing cluster cohesion. For example, SDCN [10] calculate a target distribution $P = [p_{ij}]$ as follows:

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'} q_{ij'}^2 / f_{j'}} \quad (9)$$

where $f_j = \sum_i q_{ij}$ are soft cluster frequencies. In the target distribution P , each assignment in Q is squared and normalized so that the assignments have higher confidence [10]. However, making all samples much closer to cluster centers may cause a degenerate solution that assigns all data points to a single (arbitrary) label, and thus make all samples similar and less discriminative [12]. Moreover, it makes the performance of these methods highly dependent on good initial cluster centers, thus leading to manual trial-and-error pre-training.

In order to avoid such problems, we add the constraints that the label distribution must be consistent with the mixing proportions, and we achieve such constraints through optimal transport [24]. Therefore, we construct the target probability matrix P by solving the following optimal transport strategy:

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

Here we regard the target distribution P as the transport plan matrix [24] in the optimal transport theory, and the $-\log Q$ as the cost

matrix [24] in the optimal transport theory. And we restrict $P^T \mathbf{1}_N = N\pi$, π is the proportion of points for each cluster, which can be estimated by the intermediate clustering result, thus we successfully add the constraints that the result cluster distribution must be consistent with the mixing proportions.

As directly optimizing the above objective is always time-consuming, we apply the Sinkhorn distance [46] to use an entropic constraint for fast optimization. The optimization problem with a Lagrange multiplier of the entropy constraint is as follows:

$$\begin{aligned} \min_P \quad & P * (-\log Q) - \frac{1}{\lambda} H(P) \\ \text{s.t.} \quad & P \in \mathbb{R}_+^{N \times C}, \\ & P \mathbf{1}_C = \mathbf{1}_N \quad \text{and} \quad P^T \mathbf{1}_N = N\pi \end{aligned} \quad (11)$$

where H is the entropy function and λ is the smoothness parameter controlling the equilibrium of clusters. The existence and unicity of P are guaranteed, and can be efficiently solved by Sinkhorn's [46] fixed point iteration:

$$\hat{P}^{(t)} = \text{diag}(\mathbf{u}^{(t)}) Q^\lambda \text{diag}(\mathbf{v}^{(t)}) \quad (12)$$

where t denotes the iteration and in each iteration: $\mathbf{u}^{(t)} = \mathbf{1}_N / (Q^\lambda \mathbf{v}^{(t-1)})$ and $\mathbf{v}^{(t)} = N\pi / (Q^\lambda \mathbf{u}^{(t)})$, with the initiation $\mathbf{v}^{(0)} = \mathbf{1}_N$. After the fixed point iteration, we can get the optimal transport plan matrix \hat{P} , i.e., the solution of (10).

Then during the training process, we fix \hat{P} and make Q be consistent with \hat{P} , so the graph clustering loss function can be formulated as:

$$\mathcal{L}_{GC} = \text{KL}(\hat{P} \parallel Q) = \sum_i \sum_j \hat{p}_{ij} \log \frac{\hat{p}_{ij}}{q_{ij}}. \quad (13)$$

Thus, the overall loss function is

$$\mathcal{L} = \mathcal{L}_{GE} + \gamma \mathcal{L}_{GC}, \quad (14)$$

where γ is a tuning parameter to balance the two losses, \mathcal{L}_{GE} and \mathcal{L}_{GC} .

4.3. Overall algorithm

We first train the model with cut-informed graph encoding loss. After that, we apply K-means to the embeddings to obtain the initial centroids and cluster size. Then we train the model with both graph encoding loss and graph clustering loss. Then clustering results are obtained at the last epoch. The details are summarized in the Algorithm 1.

Algorithm 1 Deep Cut-informed Graph Clustering

Require: Attribute feature matrix X ;

Graph adjacent matrix A ;

Number of clusters K ;

The number of iteration M .

Ensure: Clustering results $c = \{c_i\}_{i=1}^N$.

- 1: Calculate the graph Laplacian of original graph and attribute graph by Eq. (2) and (5).
- 2: **for** iteration from 1 to M **do**
- 3: Generate node embedding $H^{(t)}$;
- 4: Compute the graph encoding loss by Eq. (7);
- 5: Compute clustering assignments by Eq. (8) and target distribution by optimal transport strategy, i.e. (10);
- 6: Compute the graph clustering loss by Eq. (13);
- 7: Update the parameters by minimizing Eq. (14)
- 8: **end for**
- 9: Return the clustering results $c^{(M)}$.

5. Experiment

In this section, we conduct a series of experiments to validate the efficacy of our proposed DCGC framework across six datasets. The results demonstrate the superiority of our method when compared

Table 2

The statistics of the datasets.

Dataset	Network type	Nodes	Classes	Dimension
Cora	Citation	2708	7	1433
ACM	Paper	3025	3	1870
DBLP	Author	4057	4	334
Citeseer	Citation	3327	6	3703
Amazon	Item	7650	8	745
PubMed	Citation	19717	3	500

to twelve state-of-the-art approaches. We conducted sufficient ablation experiments to verify the effectiveness of each module of our framework.

5.1. Datasets

We evaluate the proposed method on six public benchmark datasets including multiple types of graphs [47]. The statistical information of these datasets is provided in Table 2 and more detailed descriptions are as follows: **Cora** is a citation network that comprises scientific publications categorized into 7 distinct classes. Each publication is represented by a binary word vector. **ACM** is a paper dataset containing papers categorized into 3 classes based on their research area. An edge indicates co-authorship of two papers. The paper features are represented by the bag-of-words of the keywords. **DBLP** is a co-authorship network where authors are categorized based on the domain of conferences to which they have contributed. The author features are represented by the bag-of-words of their keywords. **Citeseer** is a citation network where each publication within the dataset is represented by a binary word vector, with a value of 0 or 1, denoting the absence or presence. **Amazon** is a co-purchase graph of products belonging to different categories. The edges indicate two products are frequently purchased together. The features represent the product reviews encoded using a bag-of-words approach. **PubMed** is a citation network consisting of scientific publications from the PubMed database. Each publication is described by a TF/IDF weighted word vector.

5.2. Baselines

We compare the performance of our proposed method with twelve baseline methods: **K-means** [48] is a classical clustering method based on the raw data. **AE** [35] performs K-means on the representations learned by auto-encoder. **DEC** [18] employs a clustering loss to supervise the process of clustering with the auto-encoder backbone. **GAE** [49] is a framework for unsupervised learning on graph-structured data based on the variational auto-encoder. **DAEGC** [19] uses an attention network to learn the node representations and employs a clustering loss to supervise the process of graph clustering. **ARGA** [20] applies adversarial training to enforce latent code to match prior distribution. **MVGRL** [50] learns representations by contrasting encodings from first-order neighbors and a graph diffusion. **SDCN** [10] is representative of hybrid methods which take advantage of both AE and GCN modules for clustering. **AGCN** [21] utilizes the attention-based method by considering the dynamic fusion strategy and the multi-scale features fusion. **DFCN** [11] designs a dynamic cross-modality fusion mechanism and a triplet self-supervised strategy. **DRCN** [12] reduces information correlation in a dual manner to address the issue of representation collapse. **GC-SEE** [51] simultaneously considers and exploits different types of structure information to extract more comprehensive features. **CDNMF** [52] uses nonnegative matrix factorization, combined with contrastive learning between network topology and node attribute. **MGAI** [36] leverages modularity maximization as a contrastive pretext task to effectively uncover the underlying information of communities in graphs.

Table 3
Clustering results on six benchmark datasets (mean \pm std). The bold values represent the best results.

Method	Cora				ACM			
	ACC	NMI	ARI	F1	ACC	NMI	ARI	F1
K-Means	33.80 \pm 2.71	14.98 \pm 3.43	08.60 \pm 1.95	30.26 \pm 4.46	67.31 \pm 0.71	32.44 \pm 0.46	30.60 \pm 0.69	67.57 \pm 0.74
AE	49.38 \pm 0.91	25.65 \pm 0.65	21.63 \pm 0.58	43.71 \pm 1.05	81.83 \pm 0.08	49.30 \pm 0.16	54.64 \pm 0.16	82.01 \pm 0.08
DEC	46.50 \pm 0.26	23.54 \pm 0.34	15.13 \pm 0.42	39.23 \pm 0.17	84.33 \pm 0.76	54.54 \pm 1.51	60.64 \pm 1.87	84.51 \pm 0.74
GAE	43.38 \pm 2.11	28.78 \pm 2.97	16.43 \pm 1.65	33.48 \pm 3.05	84.52 \pm 1.44	55.38 \pm 1.92	59.46 \pm 3.10	57.36 \pm 0.82
DAEGC	70.43 \pm 0.36	52.89 \pm 0.69	49.63 \pm 0.43	68.27 \pm 0.57	86.94 \pm 2.83	56.18 \pm 4.15	59.35 \pm 3.89	87.07 \pm 2.79
ARGA	71.04 \pm 0.25	51.06 \pm 0.52	47.71 \pm 0.33	69.27 \pm 0.39	86.29 \pm 0.36	56.21 \pm 0.82	63.37 \pm 0.86	58.23 \pm 0.31
MVGRL	70.47 \pm 3.70	55.57 \pm 1.54	48.70 \pm 3.94	67.15 \pm 1.86	86.73 \pm 0.76	60.87 \pm 1.40	65.07 \pm 1.76	63.71 \pm 0.39
SDCN	35.60 \pm 2.83	14.28 \pm 1.91	07.78 \pm 1.24	24.37 \pm 1.04	90.45 \pm 0.18	68.31 \pm 0.25	73.91 \pm 0.40	90.42 \pm 0.19
AGCN	47.67 \pm 1.25	26.59 \pm 1.31	18.36 \pm 1.36	33.78 \pm 1.49	90.59 \pm 0.15	68.38 \pm 0.45	74.20 \pm 0.38	90.58 \pm 0.17
DFCN	36.33 \pm 0.49	19.36 \pm 0.87	04.67 \pm 2.10	26.16 \pm 0.50	90.84 \pm 0.15	69.39 \pm 0.36	74.93 \pm 0.37	90.78 \pm 0.16
DCRN	72.67 \pm 0.99	56.09 \pm 1.17	50.22 \pm 2.09	67.59 \pm 0.86	91.93 \pm 0.20	71.59 \pm 0.61	77.56 \pm 0.61	91.94 \pm 0.20
GC-SEE	73.58 \pm 0.74	53.02 \pm 0.41	51.22 \pm 0.88	71.48 \pm 0.79	91.67 \pm 0.10	70.83 \pm 0.25	76.89 \pm 0.24	91.66 \pm 0.09
CDNMF	58.59 \pm 3.44	38.74 \pm 2.57	35.72 \pm 3.51	52.69 \pm 5.32	62.94 \pm 4.67	20.91 \pm 1.77	21.21 \pm 4.58	64.56 \pm 3.48
MGAI	73.52 \pm 0.62	57.05\pm0.91	52.24 \pm 1.26	72.13 \pm 0.50	90.00 \pm 0.36	66.46 \pm 0.77	72.19 \pm 1.01	90.05 \pm 0.35
DCGC	75.41\pm1.69	56.55 \pm 2.00	53.61\pm1.94	72.15\pm3.12	92.50\pm0.32	72.98\pm0.60	78.87\pm0.63	92.51\pm0.23
Method	DBLP				PubMed			
	ACC	NMI	ARI	F1	ACC	NMI	ARI	F1
K-Means	38.65 \pm 0.65	11.45 \pm 0.38	06.97 \pm 0.39	31.92 \pm 0.27	59.83 \pm 0.01	31.05 \pm 0.02	28.10 \pm 0.01	58.88 \pm 0.01
AE	51.43 \pm 0.35	25.40 \pm 0.16	12.21 \pm 0.43	52.53 \pm 0.36	63.07 \pm 0.31	26.32 \pm 0.57	23.86 \pm 0.57	64.01 \pm 0.29
DEC	58.16 \pm 0.56	29.51 \pm 0.28	23.92 \pm 0.39	59.38 \pm 0.51	60.14 \pm 0.09	22.44 \pm 0.14	19.55 \pm 0.13	61.49 \pm 0.10
GAE	61.21 \pm 1.22	30.80 \pm 0.91	22.02 \pm 1.40	61.41 \pm 2.23	62.09 \pm 0.81	23.84 \pm 3.54	20.62 \pm 1.39	68.08 \pm 1.76
DAEGC	62.05 \pm 0.48	32.49 \pm 0.45	21.03 \pm 0.52	61.75 \pm 0.67	68.73 \pm 0.03	28.26 \pm 0.03	29.84 \pm 0.04	68.23 \pm 0.02
ARGA	64.83 \pm 0.59	29.42 \pm 0.92	27.99 \pm 0.91	64.97 \pm 0.66	65.26 \pm 0.12	24.80 \pm 0.17	24.35 \pm 0.17	65.69 \pm 0.13
MVGRL	42.73 \pm 1.02	15.41 \pm 0.63	08.22 \pm 0.50	40.52 \pm 1.51	67.01 \pm 0.52	31.59 \pm 1.45	29.42 \pm 1.06	67.07 \pm 0.36
SDCN	68.05 \pm 1.81	39.50 \pm 1.34	39.15 \pm 2.01	67.71 \pm 1.51	64.20 \pm 1.30	22.87 \pm 2.04	22.30 \pm 2.07	65.01 \pm 1.21
AGCN	73.26 \pm 0.37	39.68 \pm 0.42	42.49 \pm 0.31	72.80 \pm 0.56	64.70 \pm 0.90	26.20 \pm 2.30	24.16 \pm 2.00	52.70 \pm 1.90
DFCN	76.02 \pm 0.77	43.65 \pm 0.01	46.95 \pm 1.51	75.74 \pm 0.75	68.89 \pm 0.07	31.43 \pm 0.13	30.64 \pm 0.11	68.10 \pm 0.07
DCRN	79.66\pm0.25	48.95\pm0.44	53.60\pm0.46	79.28\pm0.26	OOM	OOM	OOM	OOM
GC-SEE	79.23 \pm 0.96	48.04 \pm 1.46	53.51 \pm 1.82	78.55 \pm 0.99	67.78 \pm 0.38	25.62 \pm 1.22	25.13 \pm 0.86	67.72 \pm 0.34
CDNMF	39.48 \pm 3.18	07.22 \pm 2.19	05.29 \pm 2.45	37.91 \pm 3.16	64.71 \pm 3.04	21.35 \pm 3.09	25.47 \pm 3.75	55.10 \pm 2.52
MGAI	77.38 \pm 0.87	48.17 \pm 0.83	49.96 \pm 1.52	77.04 \pm 0.83	62.79 \pm 1.00	22.77 \pm 1.75	18.62 \pm 1.71	62.52 \pm 0.83
DCGC	78.05 \pm 0.56	46.46 \pm 1.22	50.68 \pm 0.94	77.51 \pm 0.66	68.91\pm0.43	32.53\pm1.65	30.81\pm0.85	68.74\pm0.62
Method	Citeseer				Amazon			
	ACC	NMI	ARI	F1	ACC	NMI	ARI	F1
K-Means	39.32 \pm 3.17	16.94 \pm 3.22	13.43 \pm 3.02	36.08 \pm 3.53	43.24 \pm 4.37	30.74 \pm 4.33	17.78 \pm 2.82	30.34 \pm 7.45
AE	57.08 \pm 0.13	27.64 \pm 0.08	29.31 \pm 0.14	53.80 \pm 0.11	59.72 \pm 3.87	51.89 \pm 3.70	40.47 \pm 3.06	47.76 \pm 6.04
DEC	55.89 \pm 0.20	28.34 \pm 0.30	28.12 \pm 0.36	52.62 \pm 0.17	59.84 \pm 0.24	54.67 \pm 0.30	42.21 \pm 0.25	47.72 \pm 2.87
GAE	61.35 \pm 0.80	34.63 \pm 0.65	33.55 \pm 1.18	57.36 \pm 0.82	71.57 \pm 2.48	62.13 \pm 2.79	20.62 \pm 1.39	68.08 \pm 1.76
DAEGC	64.54 \pm 1.39	36.41 \pm 0.86	37.78 \pm 1.24	62.20 \pm 1.32	71.56 \pm 3.34	60.68 \pm 2.58	52.03 \pm 3.76	67.55 \pm 3.39
ARGA	61.07 \pm 0.49	34.40 \pm 0.71	34.32 \pm 0.70	58.23 \pm 0.31	69.28 \pm 2.30	58.36 \pm 2.76	44.18 \pm 4.41	64.30 \pm 1.95
MVGRL	68.66 \pm 0.36	43.66 \pm 0.40	44.27 \pm 0.73	63.71 \pm 0.39	53.44 \pm 0.81	36.89 \pm 1.31	29.42 \pm 1.06	39.65 \pm 2.39
SDCN	65.96 \pm 0.31	38.71 \pm 0.32	40.17 \pm 0.43	63.62 \pm 0.24	75.51 \pm 1.92	63.26 \pm 2.05	54.95 \pm 2.23	69.44 \pm 1.34
AGCN	68.79 \pm 0.23	41.54 \pm 0.30	43.79 \pm 0.31	62.37 \pm 0.20	76.80 \pm 0.40	63.17 \pm 0.72	55.67 \pm 0.84	68.32 \pm 0.62
DFCN	69.54 \pm 0.15	43.93 \pm 0.22	45.45 \pm 0.26	64.27 \pm 0.20	79.13 \pm 0.90	71.12 \pm 0.98	62.41 \pm 1.58	72.92 \pm 0.81
DCRN	70.86 \pm 0.18	45.86 \pm 0.35	47.64 \pm 0.30	65.83 \pm 0.21	79.94 \pm 0.13	73.70\pm0.24	63.69 \pm 0.20	73.82 \pm 0.12
GC-SEE	70.90 \pm 0.56	44.00 \pm 0.64	46.47 \pm 0.76	63.12 \pm 0.66	77.34 \pm 0.80	64.15 \pm 0.68	56.76 \pm 1.24	74.56 \pm 0.77
CDNMF	46.27 \pm 1.74	24.67 \pm 1.88	21.48 \pm 2.46	41.79 \pm 1.79	65.20 \pm 0.19	52.85 \pm 0.11	43.90 \pm 0.08	56.08 \pm 0.09
MGAI	70.13 \pm 0.35	44.41 \pm 0.55	46.11 \pm 0.34	65.11 \pm 0.34	78.86 \pm 0.66	69.86 \pm 0.87	61.98 \pm 0.97	72.43 \pm 0.40
DCGC	71.36\pm0.30	46.25\pm0.48	47.75\pm0.55	66.04\pm0.94	80.17\pm0.13	71.77 \pm 0.37	66.70\pm0.38	75.59\pm0.34

5.3. Metrics

To show the effectiveness of the proposed method, we employ four popular metrics [53]. 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 [54]. The four specific evaluation metrics are as follows: **ACC**: Accuracy shows the quality between the predicted labels and the true labels. It can be computed by $ACC = \frac{\sum_{n=1}^N I_n}{N}$, where I_n is an indicator function, $I_n = 1$ when the predicted label and the true label are the same, and $I_n = 0$ 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**: Macro 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 $Recall = \frac{TP}{TP + FN}$. ACC and F1 are computed after achieving the best map between the class ID and the cluster ID with the Kuhn–Munkres algorithm.

5.4. Implementation details and hyper-parameters setting

We implement our proposed model² based on PyTorch 2.0.1 and run experiments on one A100 40 GB GPU with CUDA version 12.4. A fixed MLP architecture was utilized as the encoder across all datasets to ensure fair comparison and reproducibility. The embedding size d is fixed to 16 for all datasets. We optimize DCGC with Adam optimizer with learning rate 0.005 for ACM and Citeseer, 0.001 for DBLP, Amazon, and Cora, and 0.01 for PubMed; We set the weight decay 0.005 for ACM, Citeseer, DBLP, PubMed, and Amazon, and 0.0005 for Cora. The

² The code is available at <https://github.com/CNICDS/DCGC>.

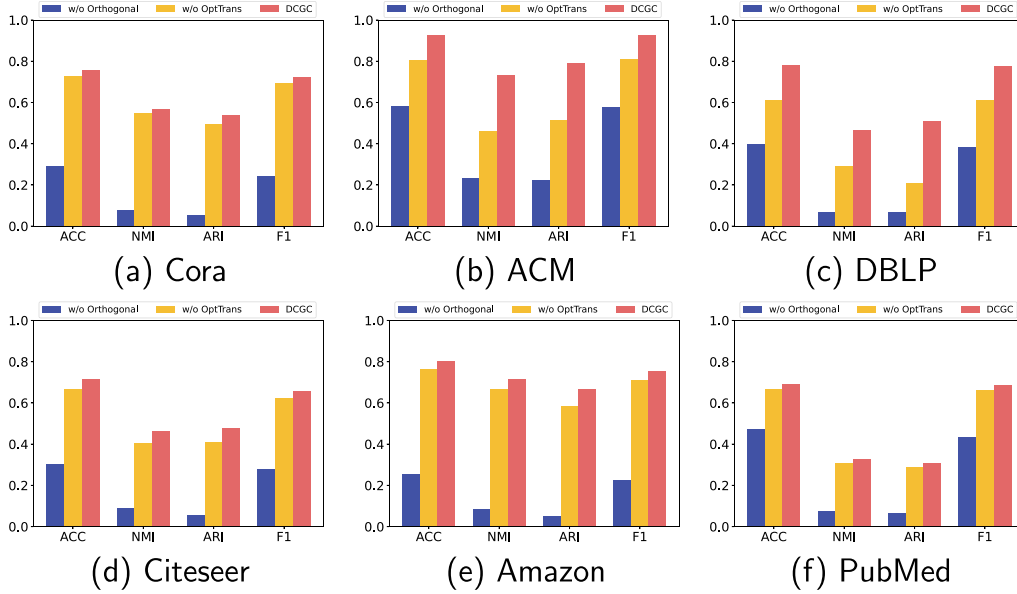


Fig. 3. Ablation comparisons of orthogonality regularization and optimal transport on six datasets.

Table 4

Ablation study on original graph and attribute graph (mean \pm std).

Dataset	Metric	DCGC - w/o OA	DCGC - w/o O	DCGC - w/o A	DCGC
Cora	ACC	36.82 \pm 2.95	46.71 \pm 1.67	73.42 \pm 2.86	75.41\pm1.69
	NMI	12.72 \pm 1.85	23.57 \pm 1.14	54.99 \pm 2.22	56.55\pm1.20
	ARI	9.53 \pm 1.81	18.54 \pm 1.21	50.44 \pm 3.24	53.61\pm1.94
	F1	35.22 \pm 3.02	43.51 \pm 1.63	69.66 \pm 4.23	72.15\pm3.12
ACM	ACC	88.79 \pm 0.17	88.95 \pm 0.75	89.97 \pm 0.92	92.50\pm0.32
	NMI	63.08 \pm 1.6	63.42 \pm 1.27	66.10 \pm 0.51	72.98\pm0.60
	ARI	69.39 \pm 2.31	69.81 \pm 1.78	72.48 \pm 1.10	78.87\pm0.63
	F1	88.76 \pm 0.91	88.90 \pm 0.82	89.99 \pm 0.62	92.51\pm0.23
DBLP	ACC	71.16 \pm 1.54	71.38 \pm 1.61	73.79 \pm 1.42	78.05\pm0.56
	NMI	37.36 \pm 3.03	37.42 \pm 1.31	41.23 \pm 1.72	46.46\pm1.22
	ARI	37.62 \pm 2.28	37.93 \pm 1.43	43.17 \pm 1.85	50.68\pm0.94
	F1	70.88 \pm 1.53	71.07 \pm 1.34	73.51 \pm 1.55	77.51\pm0.66
Citeseer	ACC	61.07 \pm 1.23	61.42 \pm 1.25	64.43 \pm 1.88	71.34\pm0.30
	NMI	34.03 \pm 1.46	34.38 \pm 1.31	39.62 \pm 1.87	46.32\pm0.48
	ARI	33.85 \pm 1.70	34.14 \pm 1.55	37.69 \pm 1.53	47.57\pm0.55
	F1	57.98 \pm 1.44	58.15 \pm 1.48	58.34 \pm 1.49	65.81\pm0.94
Amazon	ACC	52.73 \pm 1.71	54.42 \pm 1.19	75.64 \pm 1.20	80.17\pm0.13
	NMI	36.82 \pm 2.44	39.03 \pm 1.74	65.85 \pm 2.09	71.77\pm0.37
	ARI	25.46 \pm 2.44	27.33 \pm 1.34	57.19 \pm 2.64	66.70\pm0.38
	F1	51.06 \pm 1.98	51.94 \pm 1.48	70.63 \pm 1.69	75.59\pm0.34
PubMed	ACC	43.66 \pm 2.71	62.92 \pm 0.34	66.71 \pm 1.67	68.91\pm0.43
	NMI	3.37 \pm 1.67	27.69 \pm 1.41	30.16 \pm 2.90	32.53\pm1.65
	ARI	3.25 \pm 1.53	25.40 \pm 1.04	28.04 \pm 2.86	30.81\pm0.85
	F1	43.33 \pm 2.92	63.51 \pm 0.62	66.16 \pm 1.83	68.74\pm0.62

terms in the loss functions, balance parameter α is set as 0.5 for ACM, 0.4 for Citeseer, 0.8 for DBLP, and 0.9 for Amazon, PubMed and Cora; tuning parameter β is set as 2 for PubMed, 4 for ACM and Amazon, 1.5 for Citeseer, 5 for DBLP, and 20 for Cora; tuning parameter γ is set as 0.4 for ACM, 0.25 for Citeseer, 1 for DBLP, 0.5 for Amazon and PubMed, and 2 for Cora; the smoothness parameter λ is set as 20 for Amazon and PubMed, and 5 for the other four datasets; the degrees of freedom of the Student's t-distribution θ is 1 for all datasets; All hyper-parameters are achieved by the grid search method. For baseline methods compared in Table 3, we primarily utilized the hyperparameter settings reported in their original papers or the default configurations provided in their official implementations. For both stages in Section 4.3, we train the model for 100 epochs. (Taking 4 min on average.) Results are the average over 10 runs with random seeds, and we report the mean values and the corresponding standard deviations.

5.5. Overall clustering performance

The overall results of the clustering performance of all compared methods on six benchmark datasets are shown in Table 3. From these results, we can have the following observations: DCGC achieves the best results on five out of six datasets. Only on DBLP is our method sub-optimal. A possible explanation is that DBLP is not fully connected, and based on spectral embedding, DCGC may not work as expected. Compared with deep learning-based methods, our proposed DCGC avoids error message passing in GCNs or degenerated clustering results by minimizing the normalized cut and using the optimal transport theorem to guarantee the clustering process. Thus, DCGC can achieve better performance than SOTA baselines. In particular, compared with the two SOTA results of the baselines (DCRN and GC-SEE), our approach achieves a significant improvement of 0.99% on ACC, 3.79% on NMI, 1.93% on ARI, and 1.46% on F1 score, averagely.

5.6. Ablation study

We conduct ablation studies to evaluate the contributions of different components in our method.

Effect of graph information. To demonstrate both original graph and attribute graph are important, we conduct ablation studies on six datasets with four metrics. *DCGC - w/o OA* denotes the model without cut-informed graph encoding. *DCGC - w/o O*, and *DCGC - w/o A* denote the models without the original graph and the attribute graph, respectively. Table 4 shows that each graph improves the performance compared with the baseline. The results of *DCGC - w/o A* are better than those of *DCGC - w/o O* for all data sets and all metrics, indicating that the gain from introducing structural information of the graph is greater than that from introducing attribute information of the graph in the graph clustering task. And the final model DCGC with both the original graph and attribute graph performs especially better than all these baselines, which implies that the two graphs can interact to obtain better embeddings for the graph clustering task.

Effect of orthogonality regularization. From the perspective of minimizing normalized cut, the orthogonality of H is necessary. To further investigate the superiority of the proposed orthogonality regularization, we experimentally compare our method (i.e., DCGC in Fig. 3 with its baseline, i.e., w/o Orthogonal in Fig. 3). Likewise, w/o Orthogonal is

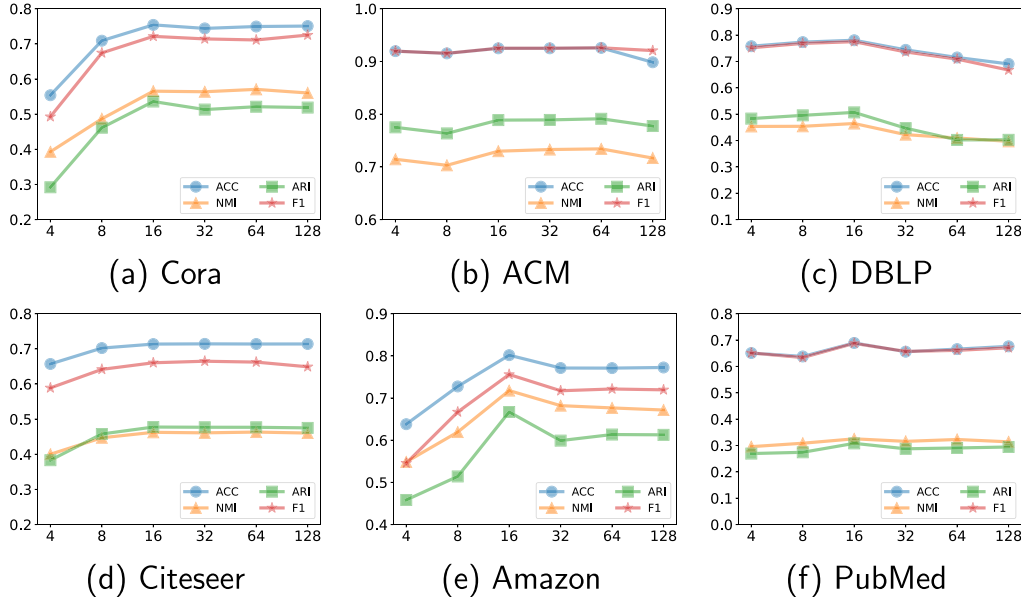


Fig. 4. Effect of dimension d on four performance metrics.

denoted that the baseline does not use the proposed orthogonality regularization. From the results in Fig. 3, we can see that DCGC outperforms w/o Orthogonal in terms of four metrics on six datasets. Such results indicate that the introduction of the orthogonality regularization in the cut-informed graph encoding module is necessary to make the learned node embeddings sufficiently different and thus avoid over-similarity and make better clustering results.

Effect of optimal transport. To demonstrate our optimal transport based self-supervised graph clustering module is better than the traditional approaches (e.g., DEC [18] and SDCN [10]) which adopt a clustering guided loss function to force the generated sample embeddings to have the minimum distortion against the pre-learned clustering center, we conduct ablation studies on six datasets with four metrics. In Fig. 3, w/o OptTrans is denoted that the baseline does not use the optimal transport based self-supervised graph clustering module but instead uses the traditional clustering guided loss function. From the results in Fig. 3, we can observe that DCGC outperforms w/o OptTrans in terms of four metrics on six datasets, which indicates that introducing optimal transport can balance the guidance of “proximity to the pre-learned cluster center”, thus avoid the degenerate solutions and achieve better results in graph clustering tasks.

Effect of embedding dimension. The graph attributes are encoded as embedding, and the dimension of the embedding can influence computational time and costs, so it is reasonable to try to find a minimal dimension that does not affect performance. To discover the effect of dimension used for embedding on metric scores, we designed a series of experiments. We varied the dimension d from 4 to 128 on the six datasets to see the clustering performance. According to the theoretical properties of normalized cut, the dimension of the embedding can be as small as the number of clusters. As shown in Fig. 4, the performance is still comparable even when $d = 4$ for ACM, DBLP, and PubMed, whose number of classes is 3, 4, and 3, respectively. As for Cora, there is a prominent rise in all 4 metrics when the dimension increases from 4 to 8, exceeding its number of classes, which is 7. Similar trends can also be observed on Citeseer and Amazon datasets. These observations are in accord with the theoretical properties of normalized cut. This implies that it is safe to use a small dimension as long as it is greater than the number of classes in the dataset.

5.7. Visualization

To show the superiority of the representation obtained by our proposed method, we use PCA to visualize the feature space. The visualizations of the clusters on ACM are given in Fig. 5. From left to right, they are the space of raw data, initialization embeddings, and learned embeddings (epoch 20 and epoch 100) of DCGC, respectively. We can see that the representations obtained by DCGC are discriminative, and each cluster is compact, indicating the clusters could be distinguished clearly in the feature space and revealing the intrinsic clustering structure among data.

5.8. Parameter sensitivity

As can be seen in Eqs. (7), (10) and (14) that DCGC introduces four hyper-parameters including balance parameter α , tuning parameter β , tuning parameter γ and the smoothness parameter λ to make a trade-off between the its modules. We conduct experiments to study the influence of these hyper-parameters on ACM datasets. Fig. 6 illustrate the effect of α , β , γ and λ varying from 0 to 1, 0.5 to 5, 0.2 to 2 and 5 to 50, respectively to study the performance variation of DCGC. From these figures, we can observe that the performance of the DCGC is stable in a wide range of hyper-parameter values.

6. Conclusion

Summary of methods. To address the issue that existing GNN-based deep graph clustering algorithms often encounter the problem of representation collapse, wherein nodes belonging to distinct categories are frequently mapped to similar representations during the sample encoding process. We propose a new and non-GNN-based graph clustering framework from the perspective of graph cut, which is more appropriate for the graph clustering task. We show that the normalized cut minimization can be relaxed into solving k -smallest eigenvalues of graph Laplacian, where the eigenvectors is a continuous approximation of graph partition and can be viewed as cut-informed graph embeddings. Specifically, we propose a cut-informed graph embedding objective to minimize the joint normalized cut of the original graph and attribute graph. And we propose a self-supervised approach via

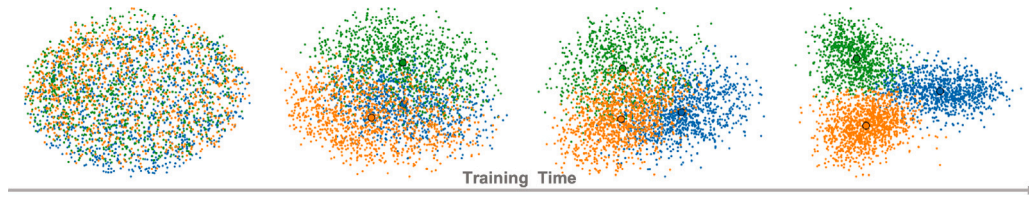


Fig. 5. PCA visualization of learned embeddings when training on the ACM dataset. Black circles indicate the cluster centroids.

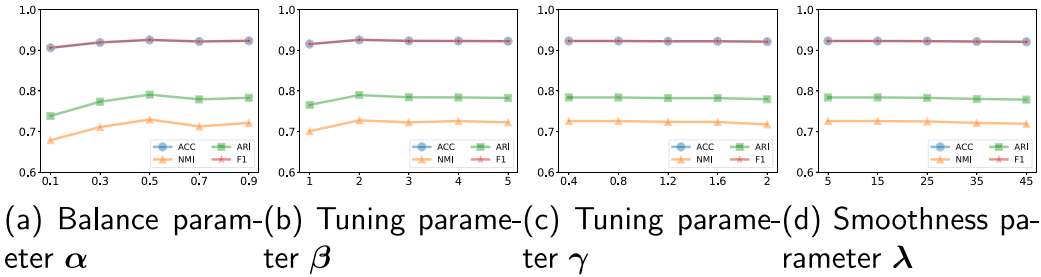


Fig. 6. Performance of DCGC on ACM benchmark datasets w.r.t. different hyper-parameters, including balance parameter α , tuning parameter β , tuning parameter γ and the smoothness parameter λ .

optimal transport to learn better clustering assignments. With the above tailored designs, our framework is more suitable for the task of graph clustering, which can effectively alleviate the problem of representation collapse and achieve better performance. Finally, we conduct extensive experiments on five challenging real-world graph datasets and show our approach can outperform the state-of-the-art deep graph clustering models.

Limitations and future work. While DCGC has shown promising results on graph clustering tasks, there are several directions for future work. First, the current framework relies on computing the eigenvectors of the graph Laplacian, which can be computationally expensive for large graphs. Investigating more scalable approximations of the normalized cut objective could allow DCGC to handle larger datasets. Investigating more advanced fusion strategies beyond the weighted sum, especially for graphs with distinct multi-modal features, remains a valuable direction for future work. Second, the optimal transport clustering module provides a self-supervised clustering approach, but further exploring semi-supervised techniques may improve performance when limited labels are available. Additionally, the effectiveness of using cosine similarity for attribute graph construction on datasets with predominantly sparse or purely categorical features requires further empirical validation. Finally, while DCGC focuses on clustering node representations, directly generating cluster assignments for new unseen nodes via inductive learning remains an open challenge. Developing inductive extensions to generalize DCGC to out-of-sample nodes would be useful for many practical graph clustering applications. In summary, potential directions include scaling to larger graphs, incorporating limited supervision, handling dynamic graphs, and inductive clustering of new nodes. Furthermore, a systematic analysis of the method's robustness against varying levels of attribute noise and informativeness was not performed and constitutes an important area for future investigation. Addressing these limitations could further enhance DCGC's modeling capabilities and applicability to real-world graph clustering tasks. But the current model provides a solid foundation and demonstrates the promise of a non-GNN cut-informed approach.

CRedit authorship contribution statement

Zhiyuan Ning: Writing – review & editing. **Zaitian Wang:** Writing – review & editing, Software, Validation. **Ran Zhang:** Writing – review & editing. **Ping Xu:** Investigation, Methodology, Software. **Kunpeng Liu:**

Investigation, Formal analysis. **Pengyang Wang:** Investigation, Formal analysis. **Wei Ju:** Project administration, Supervision. **Pengfei Wang:** Software, Methodology, Writing – original draft. **Yuanchun Zhou:** Resources, Funding acquisition, Supervision, Project administration. **Erik Cambria:** Investigation. **Chong Chen:** Investigation.

Declaration of competing interest

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

Acknowledgments

This research was supported by National Natural Science Foundation of China (Grant No. 62406306), the Science and Technology Development Fund (FDCT), Macau SAR (No. 0123/2023/RIA2, 001/2024/SKL), and the State Key Laboratory of Internet of Things for Smart City (University of Macau) No. SKL-IoTSC(UM)-2024-2026/ORP/GA02/2023.

Data availability

Data and code are open-source.

References

- [1] H. Sun, F. He, J. Huang, Y. Sun, Y. Li, C. Wang, L. He, Z. Sun, X. Jia, Network embedding for community detection in attributed networks, *ACM Trans. Knowl. Discov. from Data* 14 (2020) 1–25.
- [2] R. Hu, Z. Deng, X. Zhu, Multi-scale graph fusion for co-saliency detection, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 7789–7796.
- [3] P. Xu, Z. Ning, M. Xiao, G. Feng, X. Li, Y. Zhou, P. Wang, scCDCG: Efficient deep structural clustering for single-cell RNA-Seq via deep cut-informed graph embedding, in: *International Conference on Database Systems for Advanced Applications*, Springer, 2024, pp. 172–187.
- [4] Z. Wang, P. Wang, K. Liu, P. Wang, Y. Fu, C.-T. Lu, C.C. Aggarwal, J. Pei, Y. Zhou, A comprehensive survey on data augmentation, 2024, arXiv preprint arXiv:2405.09591.
- [5] R. Zhang, X. Wang, S. Wang, K. Liu, Y. Zhou, P. Wang, H2D: Hierarchical heterogeneous graph learning framework for drug-drug interaction prediction, in: *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 4283–4287.

- [6] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: *Proceedings of the International Conference on Learning Representations, ICLR*, 2017.
- [7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: *Proceedings of the International Conference on Learning Representations, ICLR*, 2018.
- [8] Z. Ning, Z. Qiao, H. Dong, Y. Du, Y. Zhou, Lightcake: A lightweight framework for context-aware knowledge graph embedding, in: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2021, pp. 181–193.
- [9] X. Guo, L. Gao, X. Liu, J. Yin, Improved deep embedded clustering with local structure preservation, in: *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*, 2017, pp. 1753–1759.
- [10] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, P. Cui, Structural deep clustering network, in: *Proceedings of the World Wide Web Conference, WWW*, 2020, pp. 1400–1410.
- [11] W. Tu, S. Zhou, X. Liu, X. Guo, Z. Cai, E. Zhu, J. Cheng, Deep fusion clustering network, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 9978–9987.
- [12] Y. Liu, W. Tu, S. Zhou, X. Liu, L. Song, X. Yang, E. Zhu, Deep graph clustering via dual correlation reduction, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 7603–7611.
- [13] Z. Ning, P. Wang, P. Wang, Z. Qiao, W. Fan, D. Zhang, Y. Du, Y. Zhou, Graph soft-contrastive learning via neighborhood ranking, 2022, arXiv preprint arXiv:2209.13964.
- [14] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, D. Koutra, Beyond homophily in graph neural networks: Current limitations and effective designs, *Adv. Neural Inf. Process. Syst.* 33 (2020) 7793–7804.
- [15] J. Zhu, R.A. Rossi, A. Rao, T. Mai, N. Lipka, N.K. Ahmed, D. Koutra, Graph neural networks with heterophily, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11168–11176.
- [16] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, X. Sun, Measuring and relieving the over-smoothing problem for graph neural networks from the topological view, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3438–3445.
- [17] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: *International Conference on Machine Learning*, PMLR, 2017, pp. 1263–1272.
- [18] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: *Proceedings of the International Conference on Machine Learning, ICML*, 2016, pp. 478–487.
- [19] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, C. Zhang, Attributed graph clustering: A deep attentional embedding approach, in: *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*, 2019, pp. 3670–3676.
- [20] S. Pan, R. Hu, S.-F. Fung, G. Long, J. Jiang, C. Zhang, Learning graph embedding with adversarial training methods, *IEEE Trans. Cybern.* 50 (6) (2020) 2475–2487.
- [21] Z. Peng, H. Liu, Y. Jia, J. Hou, Attention-driven graph clustering network, 2021, arXiv preprint arXiv:2108.05499.
- [22] J. Shi, J. Malik, Normalized cuts and image segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.* 22 (8) (2000) 888–905.
- [23] N. Binkiewicz, J.T. Vogelstein, K. Rohe, Covariate-assisted spectral clustering, *Biometrika* 104 (2) (2017) 361–377.
- [24] G. Monge, Mémoire sur la théorie des déblais et des remblais, *Mem. Math. Phys. Acad. R. Sci.* (1781) 666–704.
- [25] R. Zhang, X. Wang, K. Liu, Y. Zhou, P. Wang, M2mol: Multi-view multi-granularity molecular representation learning for property prediction, in: *International Conference on Database Systems for Advanced Applications*, Springer, 2024, pp. 264–274.
- [26] Y. Zhou, P. Wang, H. Dong, D. Zhang, D. Yang, Y. Fu, P. Wang, Make graph neural networks great again: a generic integration paradigm of topology-free patterns for traffic speed prediction, in: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, 2024, pp. 2607–2615.
- [27] R. Zhang, X. Wang, P. Wang, Z. Meng, W. Cui, Y. Zhou, HTCL-DDI: a hierarchical triple-view contrastive learning framework for drug-drug interaction prediction, *Brief. Bioinform.* 24 (6) (2023) bbad324.
- [28] P. Le-Hong, E. Cambria, Integrating graph embedding and neural models for improving transition-based dependency parsing, *Neural Comput. Appl.* 36 (6) (2023) 2999–3016, <http://dx.doi.org/10.1007/s00521-023-09223-3>.
- [29] H. Gao, H. Huang, Deep attributed network embedding, in: *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*, 2018, pp. 3364–3370.
- [30] M. Liu, J. Liu, Y. Dong, R. Mao, E. Cambria, Interest-driven community detection on attributed heterogeneous information networks, *Inf. Fusion (ISSN: 1566-2535)* 111 (2024) 102525.
- [31] R. Zhang, X. Wang, G. Liu, P. Wang, Y. Zhou, P. Wang, Motif-oriented representation learning with topology refinement for drug-drug interaction prediction, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 1, 2025, pp. 1102–1110.
- [32] C. Yang, Z. Liu, D. Zhao, M. Sun, E. Chang, Network representation learning with rich text information, in: *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*, 2015, pp. 2111–2117.
- [33] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, H. Liu, Attributed network embedding for learning in a dynamic environment, in: *Proceedings of the International Conference on Information and Knowledge Management, CIKM*, 2017, pp. 387–396.
- [34] X. Huang, J. Li, X. Hu, Label informed attributed network embedding, in: *Proceedings of the ACM International Conference on Web Search and Data Mining, WSDM*, 2017, pp. 731–739.
- [35] G. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *Science* 313 (2006) 504–507.
- [36] Y. Liu, J. Li, Y. Chen, R. Wu, E. Wang, J. Zhou, S. Tian, S. Shen, X. Fu, C. Meng, et al., Revisiting modularity maximization for graph clustering: A contrastive learning perspective, in: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 1968–1979.
- [37] G. Peyré, M. Cuturi, et al., Computational optimal transport, *Cent. Res. Econ. Stat. Work. Pap.* (2017–86) (2017).
- [38] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: *International Conference on Machine Learning, PMLR*, 2017, pp. 214–223.
- [39] L. Chen, Y. Zhang, R. Zhang, C. Tao, Z. Gan, H. Zhang, B. Li, D. Shen, C. Chen, L. Carin, Improving sequence-to-sequence learning via optimal transport, 2019, arXiv preprint arXiv:1901.06283.
- [40] C. Laclau, I. Redko, B. Matei, Y. Bennani, V. Brault, Co-clustering through optimal transport, in: *International Conference on Machine Learning, PMLR*, 2017, pp. 1955–1964.
- [41] K. Grauman, T. Darrell, Fast contour matching using approximate earth mover's distance, in: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004. CVPR 2004, vol. 1, IEEE, 2004, 1–I.
- [42] M. Cuturi, Sinkhorn distances: Lightspeed computation of optimal transport, in: *Proceedings of the International Conference on Neural Information Processing Systems, NeurIPS*, 2013, pp. 2292–2300.
- [43] M. Kusner, Y. Sun, N. Kolkin, K. Weinberger, From word embeddings to document distances, in: *International Conference on Machine Learning, PMLR*, 2015, pp. 957–966.
- [44] L. Wu, I.E. Yen, K. Xu, F. Xu, A. Balakrishnan, P.-Y. Chen, P. Ravikumar, M.J. Witbrock, Word mover's embedding: From word2vec to document embedding, 2018, arXiv preprint arXiv:1811.01713.
- [45] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, *J. Mach. Learn. Res.* 9 (11) (2008).
- [46] R. Sinkhorn, Diagonal equivalence to matrices with prescribed row and column sums, *Am. Math. Mon.* 74 (1967) 402–405.
- [47] L. Yue, X. Jun, Z. Sihang, W. Siwei, G. Xifeng, Y. Xihong, L. Ke, T. Wenxuan, L.X. Wang, et al., A survey of deep graph clustering: Taxonomy, challenge, and application, 2022, arXiv preprint arXiv:2211.12875.
- [48] J.A. Hartigan, M.A. Wong, A K-means clustering algorithm, *Appl. Stats* 28 (1979) 100–108.
- [49] T.N. Kipf, M. Welling, Variational graph auto-encoders, 2016, ArXiv Abs/1611.07308.
- [50] K. Hassani, A.H. Khasahmadi, Contrastive multi-view representation learning on graphs, in: *International Conference on Machine Learning, PMLR*, 2020, pp. 4116–4126.
- [51] S. Ding, B. Wu, X. Xu, L. Guo, L. Ding, Graph clustering network with structure embedding enhanced, *Pattern Recognit.* 144 (2023) 109833.
- [52] Y. Li, J. Chen, C. Chen, L. Yang, Z. Zheng, Contrastive deep nonnegative matrix factorization for community detection, in: *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, IEEE, 2024, pp. 6725–6729.
- [53] R. Xia, Y. Pan, L. Du, J. Yin, Robust multi-view spectral clustering via low-rank and sparse decomposition, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, 2014, pp. 2149–2155.
- [54] L. Lovász, M.D. Plummer, Matching Theory, vol. 29, in: *Annals of Discrete Mathematics*, North-Holland, Amsterdam, 1986.