DGCLUSTER: A Neural Framework for Attributed Graph Clustering via Modularity Maximization

Aritra Bhowmick[1], **Mert Kosan**[2*], **Zexi Huang**[2], **Ambuj Singh**[2], **Sourav Medya**[3]
[1]New York University
[2]University of California, Santa Barbara
[3]University of Illinois, Chicago
aritra.bhowmick@nyu.edu, mertkosan@gmail.com, zexihuang.phd@gmail.com, ambuj@ucsb.edu, medya@uic.edu

Abstract

Graph clustering is a fundamental and challenging task in the field of graph mining where the objective is to group the nodes into clusters taking into consideration the topology of the graph. It has several applications in diverse domains spanning social network analysis, recommender systems, computer vision, and bioinformatics. In this work, we propose a novel method, DGCLUSTER, which primarily optimizes the modularity objective using graph neural networks and scales linearly with the graph size. Our method does not require the number of clusters to be specified as a part of the input and can also leverage the availability of auxiliary node level information. We extensively test DGCLUSTER on several real-world datasets of varying sizes, across multiple popular cluster quality metrics. Our approach consistently outperforms the state-of-the-art methods, demonstrating significant performance gains in almost all settings.

Introduction and Related Work

Graph clustering is a fundamental problem in network analysis and plays an important role in uncovering structures and relationships between the nodes or entities in a graph. It has numerous applications in several domains such as community detection in social networks (**?**), identifying functional modules in biological systems (**?**), image segmentation in computer vision (**?**), and recommender systems (**?**). The primary goal of graph clustering is to group nodes with similar characteristics or functions while maintaining a clear distinction between different clusters.

Node attributes. While traditional graph clustering methods primarily rely on graph topology such as modularity maximization (**??**), recent research (**?**) has recognized the importance of incorporating node attributes in the clustering process, offering a more comprehensive approach for grouping nodes. Node attributes provide additional information associated with each node and provide contextual insights that can improve the accuracy in the clustering process.

GNN-based clustering. Recently, there have been several attempts that use the power of deep learning in the form of Graph neural networks (GNNs) (**???**) for graph clustering. GNNs provide a powerful tool in graph-based machine learning that is successful in many diverse prediction tasks (**???**), by incorporating the graph topology node features or attributes. For GNN-based graph clustering, MGAE (**?**) marginalizes the corrupted node features to learn representations via a graph encoder and applies spectral clustering. Another graph autoencoder based approach has been proposed in (**?**). To improve the efficiency of clustering, contrastive learning methods have been used recently as well (**???**). For more neural methods on deep graph clustering, we refer the readers to this recent survey (**?**).

Neural modularity maximization. One of the initial methods to optimize modularity via deep learning for graph clustering is proposed by **?**. They design a nonlinear reconstruction method based on graph autoencoders, which also incorporate constraints among node pairs. **?** propose a method that obtains a spatial proximity matrix by using the adjacency matrix and the opinion leaders in the social network. The spatial eigenvectors of the proximity matrix are applied subsequently to optimize modularity. **?** is another study that incorporates the modularity metric for community detection and graph clustering. **??** try to find communities without predefined community structure. **?** propose a generative model for community detection using a variational autoencoder. **?** firstly analyze the possible number of communities in the graph, then fine-tune it using modularity. Later on, **?** use a graph neural network that optimizes modularity and attributes similarity objectives. Another related work in this domain is the method DMoN by **?**. This method designs an architecture to encode cluster assignments and then formulate a modularity-based objective function for optimizing these assignments.

Our contributions. A vast majority of these methods have the limitation that they require the number of clusters to be given as an input or they do not take full advantage of the associated node attributes along with the additional node level information like partial availability of labels or samples of known pairwise memberships. To overcome these challenges, we propose a framework named Deep Graph Cluster (DGCLUSTER) that eliminates the need for a predefined number of clusters and harnesses graph representation learning methods that can leverage node

[*]Work done prior to joining Visa Research.

attributes along with other available auxiliary information. Our major contributions are as follows.

- **DGCLUSTER:** We develop a novel framework that uses pairwise (soft) memberships between nodes to solve the graph clustering problem via modularity maximization. The complexity of our framework scales linearly with the size of the graph.
- **Handling Unknown Number of Clusters and Auxiliary Information:** Our proposed methodology can generalize well to cases when the number of clusters is not known beforehand. Our designed loss function is also flexible towards accommodating the additional local or node-level information. These are the major strengths of our approach.
- **Extensive Empirical Evaluation:** We conduct extensive experiments on seven real-world datasets of different sizes on four different objectives that quantify the quality of clusters. Our method shows significant performance gain against state-of-the-art methods in most of the settings.

Problem Definition

We consider an undirected and unweighted graph $G = (V, E)$, where $V = \{v_1, v_2, v_3, ..., v_n\}$ is the set of n vertices/nodes, and $E = \{e_{ij} = (v_i, v_j)\}$ is the set of m edges. The adjacency matrix of G can be represented as a non-negative symmetric matrix $A = [A_{ij}] \in R_+^{n\times n}$ where $A_{ij} = 1$ if there is an edge between vertices i and j, and $A_{ij} = 0$. The degree of vertex i is defined as $d_i = \sum_j a_{ij}$. In addition, we have features (or attributes) associated with each node in the graph, $X^0 \in \mathbb{R}^{n\times r}$ where r is the size of the feature vector on nodes.

Graph clustering. Our objective is to have a disjoint clustering of the nodes in the graph. More specifically, the problem of *graph clustering* is to partition the set of nodes into k clusters or groups $\{V_i\}_{i=1}^k$ ($V_i \cap V_j = \emptyset$ for $i \neq j$), such that the nodes within a cluster are more densely connected than nodes belonging to different clusters. Furthermore, in this work, we aim to incorporate node attributes in addition to the graph topology for graph clustering.

While there exist several metrics to measure the quality of clustering such as conductance (**?**) and normalized cut-ratio (**?**), modularity remains the most popular and widely used metric for graph clustering in the literature (**?**).

Modularity. The approach of graph clustering based on maximizing the modularity of the graph has been introduced by Newman (**?**). As a graph topology-based measure, modularity (**?**) quantifies the difference between the fraction of the edges that fall within the clusters and the expected fraction assuming the edges have been distributed randomly. Formally, modularity (Q) is defined as follows:

$$Q = \frac{1}{2m}\sum_{ij}(A_{ij} - \frac{d_i d_j}{2m})\delta(c_i, c_j) \tag{1}$$

where $\delta(c_i, c_j)$ is the Kronecker delta, i.e., $\delta(c_i, c_j) = 1$ if $c_i = c_j$ and 0 otherwise, and c_i is the community to which node i is assigned.

The value of modularity for unweighted and undirected graphs lies in the range $[-1/2, 1]$. The Q value close to 0 implies that the fraction of edges inside communities is no better than a random distribution, and higher values usually correspond to a stronger cluster structure. The modularity Q can also be expressed in the matrix form as follows:

$$Q = \frac{1}{2m}\sum_{ij} B \odot M = \frac{1}{2m}Tr(BM^T) = \frac{1}{2m}Tr(BM) \tag{2}$$

where M is a $n \times n$ symmetric matrix with $M_{ij} = \delta(c_i, c_j)$ and $B_{ij} = (A_{ij} - \frac{d_i d_j}{2m})$ is called the modularity matrix.

Modularity maximization. Since a larger Q implies a prominent cluster structure, optimizing the modularity is a popular way of finding good clusters. While modularity optimization is known to be NP-Hard (**?**), there exist techniques such as spectral relaxation and greedy algorithms, which permit efficient solutions (**??**).

Our goal. We achieve graph clustering via modularity maximization. The definition of modularity brings the idea of computing pairwise memberships allowing a natural interpretation without knowing the number of clusters. We aim to take advantage of that and the power of graph representation learning techniques that can exploit both structural and non-structural information from graphs. In the experiments, we show the efficacy of our method on four different objectives that quantify the quality of the clusters: modularity (**?**), conductance (**?**), Normalized mutual information (NMI), and F1 score.

Method: DGCLUSTER

We present DGCLUSTER, a fully differentiable method which performs deep graph clustering based on the graph structure and node attributes, without the need to predefine the number of communities. The key rationale of our method is to parameterize a relevant clustering objective (e.g., Modularity) with similarity between nodes computed based on graph neural network (GNN)-based embeddings. Our method DGCLUSTER consists of four steps:

- **Node embeddings.** As the first step, DGCLUSTER obtains the node embeddings as the output of the GNN followed by some transformations that helps to perform efficient clustering.
- **Modularity via similarity.** We evaluate the similarity between all node pairs from the embeddings and treat them as soft community pairwise memberships.
- **Objectives.** Subsequently, it builds the cluster detection objectives based on the soft memberships and train the GNN parameters in a differentiable manner.
- **Final clustering.** Finally, it computes the community memberships by clustering the GNN node embeddings.

We introduce each step of our method DGCLUSTER in the following subsections.

Node embeddings using GNN

We begin with a brief introduction of graph neural networks (GNNs). GNNs are powerful graph machine learning paradigms that combine the graph structure and node attribute information into node embeddings for different downstream tasks. A key design element of GNNs is message passing where the nodes iteratively update their representations (embeddings) by aggregating information from their neighbors. In the literature, several different GNN architectures have been proposed (**????**) which implement different schemes of message passing. A comprehensive discussion on the methods and applications of GNNs are described here (**?**).

In this paper, we leverage the widely used Graph Convolutional Network (GCN) (**?**) to produce node embeddings, noting that our model can be equipped with other GNNs. With the initial node features as $X^{(0)}$, the layer-wise message passing rule for layer l ($l = 0, \cdots, L-1$) is as follows:

$$X^{(l+1)} = \sigma(\tilde{A}X^{(l)}W^{(l)}) \tag{3}$$

where $\tilde{A} = D^{-\frac{1}{2}}AD^{\frac{1}{2}}$ is the normalized adjacency matrix, D is the diagonal node degree matrix, X^l is the embedding output of the l-th layer, W^l is the learnable weight matrix of the lth layer, and σ is the activation function which introduces the non-linearity in the feature aggregation scheme. We do not use any self loop creation in the adjacency matrix and we choose the SELU (**?**) as the activation function for better convergence. The SELU activation is given as:

$$SELU(x) = \begin{cases} \beta x, & \text{if } x \geq 0 \\ \beta\alpha(e^x - 1), & \text{otherwise} \end{cases} \tag{4}$$

where $\beta \approx 1.05$ and $\alpha \approx 1.67$.

Transformation of the embeddings. Let $X = X^L = [X_1, \cdots, X_n]^T$ be the output embeddings of the last layer readout from the GNN. We introduce the following problem-specific transformations on the embeddings (where $Z^{\circ 2}$ denotes the element-wise square operation):

$$\begin{aligned} X_i &\leftarrow \frac{X_i}{\sum_j X_{ij}}, \\ X_i &\leftarrow \tanh(X_i), \\ X_i &\leftarrow X_i^{\circ 2}, \\ X_i &\leftarrow \frac{X_i}{\|X_i\|_2} \end{aligned} \tag{5}$$

Specifically, the first normalization is used so as to prevent vanishing gradients because of the $\tanh$ activation function for large values. Next, after the activation, doing the element-wise square ensures the output is constrained within the positive coordinate space. The final L_2 normalization reduces the cosine similarity computation of node pairs (introduced in next subsection) to corresponding dot products. Thus, the final embeddings lie on the surface of the unit sphere constrained in the positive space. More detailed intuitions behind these will be explained in details in the following sections.

Modularity via embedding similarity

After obtaining the transformed GNN embeddings X, we demonstrate how to compute the modularity Q based on them via pairwise node similarities.

To achieve this, we focus on M which is defined in Eq. **??** as a binary matrix that encodes the pairwise memberships (via δ) of the nodes in a cluster. This pairwise relationship is transitive, i.e., $\delta(c_u, c_v) = 1$ and $\delta(c_v, c_w) = 1$ implies $\delta(c_u, c_w) = 1$. However, as stated before, the problem of finding the optimal M which maximizes Q is NP-Hard. The main idea is to replace M with a soft pairwise community membership matrix. We choose to replace M with a similarity matrix which is defined based on node embeddings, where the similarity ($f_{sim}(X_u, X_v) \in [0, 1]$) can be viewed as soft membership between the nodes $\{u, v\}$. Higher values of $f_{sim}(X_u, X_v)$ corresponds to higher similarity or stronger relationship between the nodes. Although there can be many choices for the similarity function, we choose f_{sim} as the cosine similarity, $f_{sim}(X_u, X_v) = \cos(X_u, X_v)$.

Here, we also emphasize our rationale for the transformations of the embeddings earlier. Specifically, the original M only takes binary values (i.e., 0 or 1). Our embedding transformation allows $\cos(X_u, X_v) \in [0, 1]$ by limiting them in the positive coordinate space, and enables its computation via dot products $\cos(X_u, X_v) = X_u^T X_v$, which in turn leads to an efficient computation as discussed later.

Objective function

We introduce a novel joint objective function that performs clustering based on both the community structure quality measure (i.e., modularity) and local auxiliary information.

Modularity optimization. We first define our primary objective function, i.e., modularity optimization by approximating the pairwise community membership computed with embedding similarity:

$$L_1 = -\tilde{Q} = -\frac{1}{2m}Tr(BXX^T) \tag{6}$$

where B is the modularity matrix, m is the number of edges, and XX^T is the similarity matrix obtained from the transformed node embeddings.

Auxiliary information loss. We now discuss how our algorithm can be made more flexible by accounting for additional local information.

Specifically, let $S \subseteq V$ be the subset of nodes whose local information is available, and $H \in \mathbb{R}^{|S|\times|S|}$ be the pairwise information matrix. We consider different types of additional information. In the semi-supervised setting, partial node labels (e.g. cluster labels, class labels) are available, and we can construct H based on the pairwise membership:

$$H_{ij} = \begin{cases} 1, & \text{if } c_i = c_j \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where c_i is the label of the node i. Alternatively, by collecting all available labels in one-hot matrix form $C \in \mathbb{R}^{|S|\times p}$, we can rewrite H as:

$$H = CC^T \tag{8}$$

When those ground-truth information is not available, we can also leverage traditional structure-based graph partitioning heuristics, such as Louvain (**?**), and treat their generated clusters as the node labels. In general, any pairwise node information similarity which can be approximated as $\langle C_i, C_j \rangle \in [0, 1]$, can be effectively used.

With the local information matrix H, the secondary objective function which minimizes the difference between H and the embedding similarity matrix is given as :

$$L_2 = \frac{1}{|S|^2} \|H - X_S X_S^T\|_F^2 \tag{9}$$

where X_S is the submatrix of node embeddings with only nodes in S.

The final objective function is a weighted combination of the two objectives:

$$L = L_1 + \lambda L_2 \tag{10}$$

where λ is a hyperparameter. The GNN parameters are trained in an end-to-end based on the total loss L with the stochastic gradient descent algorithm.

When instead of individual node labels, samples of pairwise node memberships are available, L_2 can be written as follows:

$$L_2 = \frac{1}{|S|} \sum_{p_{ij} \in S} (1 - \langle X_i, X_j \rangle)^2 \tag{11}$$

where p_{ij} are the node pairs which belong to the same community and S is a set of such pairs.

Clustering node embeddings

In this section, we illustrate how to obtain the hard cluster partitions based on the soft pairwise memberships obtained in the previous section.

One way is to directly apply clustering algorithms based on the pairwise node similarity matrix, such as affinity propagation (**?**). However, computing the full similarity matrix from the embeddings is computationally prohibitive. Instead, we take advantage of the following observation. Since our embeddings are L_2 normalized (i.e., $\|X_u\|_2 = 1$), the cosine similarity is directly related to the Euclidean distance in the embedding space:

$$\begin{aligned} \|X_u - X_v\|_2^2 &= \|X_u\|_2^2 + \|X_v\|_2^2 - 2\|X_u\|_2 \|X_v\|_2 \cos(X_u, X_v) \\ &= 2(1 - \cos(X_u, X_v)) \end{aligned} \tag{12}$$

This allows us to apply clustering algorithms based on the euclidean distance in the embedding space without computing the full pairwise similarity matrix.

Specifically, we apply the Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) (**?**). BIRCH is a scalable, memory-efficient, online clustering algorithm that can cluster large datasets by first generating a small and compact summary of the dataset that retains as much information as possible. Unlike other popular choices such as k-means (**?**), BIRCH does not require the number of clusters beforehand.

Complexity analysis

In this section, we analyze the complexity of our proposed model. The forward pass requires us to compute two objective functions. Specifically, modularity optimization loss L_1 can be evaluated with

$$\begin{aligned} L_1 &= -\frac{1}{2m} Tr(BXX^T) = -\frac{1}{2m} Tr(X^T BX) \\ &= -\frac{1}{2m} (Tr(X^T AX) - \frac{1}{2m} Tr(X^T dd^T X)) \end{aligned} \tag{13}$$

We can see that L_1 can be computed with sparse matrix multiplications between X and A and matrix vector multiplications between X and d. These multiplications lead to an overall computation cost of $O(k^2 n)$, where k is the dimension of the embeddings X.

For the auxiliary information loss, we have

$$\begin{aligned} &\|H - X_S X_S^T\|_F^2 = \sum_{ij} (H_{ij} - (XX^T)_{ij})^2 \\ &= \sum_{ij} H_{ij}^2 + \sum_{ij} (XX^T)_{ij}^2 - 2 \sum_{ij} H_{ij} (XX^T)_{ij} \\ &= \sum_{ij} (CC^T)_{ij}^2 + \sum_{ij} (XX^T)_{ij}^2 - 2 \sum_{ij} (CC^T)_{ij} (XX^T)_{ij} \\ &= Tr(C^T CC^T C) - Tr(X_S^T X_S X_S^T X_S) - 2Tr(X_S^T CC^T X_S) \end{aligned} \tag{14}$$

Computing $C^T C$, $X_S^T X_S$, and $X_S^T C$ requires $O(p^2 n)$, $O(k^2 n)$, and $O(knp)$ respectively via matrix multiplications, assuming $|S| = n$, since $|S|$ can be atmost n. Here, p is the dimension of the auxiliary information (note, $C \in \mathbb{R}^{n \times p}$). Thus, the overall complexity of our model is $O(k^2 n + p^2 n)$, where $k, p \ll n$. This shows that our model scales linearly with the size of the graph.

Experimental Results

In this section, we present the performance comparison of DGCLUSTER with other clustering and neural community detection methods on 7 *well-known real-world datasets*. We evaluate the algorithms using various metrics, including *graph conductance, modularity, NMI (Normalized Mutual Information), and F1 score* where higher scores are desired for the last three. Furthermore, we demonstrate the robustness of our method against different GNN architectures and assess the impact of adding auxiliary information. We also introduce an additional regularization objective and show its effect on our method. Lastly, we illustrate the number of communities our method identifies for each dataset. Our code and implementation of DGCLUSTER is available at https://github.com/pyrobits/DGCluster

Datasets. We use seven publicly available real-world datasets in our experiments. Table **??** presents a summary of the dataset statistics. Our experiments include datasets from well-known citation networks, namely CORA, CITESEER, and PUBMED (**?**). In these networks, nodes correspond to individual papers, edges represent citations between papers, and features are extracted using a bag-of-words approach applied to paper abstracts. The topic of each paper is captured through node labels. Besides citation networks, we use two co-purchase networks, AMAZON PC and AMAZON PHOTO (**?**). These networks include products as nodes, with edges denoting co-purchase relationships. Features are extracted from product reviews, while node labels show product categories. Our last two datasets, COAUTHOR CS and COAUTHOR PHY (**??**), are co-authorship networks for computer science and physics, respectively. Within these networks, nodes correspond to authors, and edges indicate co-authorship between them. Node features are keywords extracted from authors' publications and node labels are their fields of study.

	$\|V\|$	$\|E\|$	$\|X\|$	$\|Y\|$
CORA	2708	5278	1433	7
CITESEER	3327	4552	3703	6
PUBMED	19717	44324	500	3
AMAZON PC	13752	245861	767	10
AMAZON PHOTO	7650	119081	745	8
COAUTHOR CS	18333	81894	6805	15
COAUTHOR PHY	34493	247962	8415	5

Table 1: Statistics of the datasets. $|V|$, $|E|$, $|X|$, and $|Y|$ denote the number of nodes, edges, features, and node labels.

Baselines. Our baseline methods consist of a range of classical clustering algorithms, such as k-means, as well as state-of-the-art graph community detection algorithms. For consistent comparison, we adopted the same baseline setting as in the recent **DMoN** (**?**): **k-means** based on features, **SBM** by **?**, **k-means** based on DeepWalk by **?**, **k-means**(DGI) by **?**, **AGC** by **?**, **DAEGC** by **?**, **SDCN** by **?**, **NOCD** by **?**, **DiffPool** by **?**, **MinCutPool** by **?**, and **Ortho** by **?**.

Performance Measures. We use our primary objective, modularity, as our main evaluation metric. Additionally, we evaluate the performance using other important metrics, namely NMI, conductance, and F1 score. In cases where a dataset lacks ground truth cluster labels, we assign node labels as their respective cluster labels. We multiply the metrics by 100 for better readability.

- Modularity (Q) (**?**): This serves as the primary objective, aiming to quantify the strength of the community. It achieves this by contrasting the proportion of edges within the community with a corresponding fraction generated from random connections between nodes. Higher values mean better community partitions.
- Conductance (C) (**?**): The graph's conductance measures the portion of total edges that goes outside the community. Lower values indicate better community partitions.
- Normalized mutual information (NMI): We use the NMI as a label-based metric between the cluster assignments and true labels of the nodes. Higher values mean better community partitions.
- F1 score: We calculate the pairwise F1 score based on found pairwise node memberships and their corresponding clusters. Since it is $O(N^2)$, we sample 1000 nodes to calculate this. Higher values indicate better community partitions.

Other Settings. In our experiments, we employ a GCN having two hidden layers of 256 and 128 nodes and an output dimension of 64, across all datasets. We showcase outcomes for various λ values. This enables a comparative analysis among the graph structure-based metrics and graph attribute-based metrics. We use Adam optimizer with a learning rate set to 0.001, and we set the number of epochs to 300. Our tables and figures report average scores over 10 different runs, using different seeds, for our method. For baselines, we report the results from **?**.

Performance

Table **??** and **??** show that our method achieves superior or comparable results compared to the baselines across all evaluation metrics and datasets. Notably, our method stands out with significant improvements in unsupervised graph evaluation topology-based metrics such as graph conductance, and modularity (Table **??**). Our auxiliary objective also provides flexibility by tuning the hyperparameter λ to optimize NMI and F1 score to have better results compared to the baselines (Table **??**). While our method has a superior performance in co-purchase and co-authorship networks under all four metrics, it has a reasonably good performance in the citation networks. This shows the generalizability of our algorithm in different types of networks.

Robustness on Different Base GNNs

We conduct experiments to test the robustness of our method towards different GNNs including GAT (**?**), GIN (**?**), and GraphSAGE (**?**). Table **??** indicates that the selection of the GNN model does not significantly alter the performance of our method in terms of the evaluation measures. The average performance across datasets for different metrics and GNN bases reveals specific enhancements: GAT improves Q, GraphSAGE enhances C and F1 scores on average. Conversely, GIN results into general performance degradation.

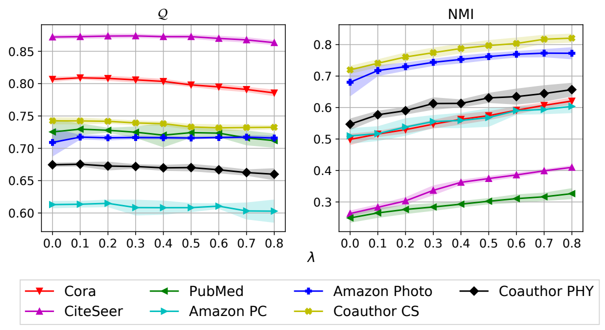

Figure 1: Effect of auxiliary information on different metrics and datasets with varying λ. As expected, Q tends to decrease as λ increases since Q is dependent only on the graph structure whereas λ adds weight to the label-based loss. The NMI increases as λ increases since NMI is directly related to labels.

	Cora		Citeseer		Pubmed		Amazon PC		Amazon Photo		Coauthor CS		Coauthor PHY	
method	$C \downarrow$	$Q \uparrow$	$C \downarrow$	$Q \uparrow$	$C \downarrow$	$Q \uparrow$	$C \downarrow$	$Q \uparrow$	$C \downarrow$	$Q \uparrow$	$C \downarrow$	$Q \uparrow$	$C \downarrow$	$Q \uparrow$
k-m(feat)	61.7	19.8	60.5	30.3	55.8	33.4	84.5	5.4	79.6	10.5	49.1	23.1	57.0	19.4
SBM	15.4	77.3	14.2	78.1	39.0	53.5	31.0	60.8	18.6	**72.2**	20.3	72.7	25.9	66.9
k-m(DW)	62.1	30.7	68.1	24.3	**16.6**	**75.3**	67.6	11.8	60.6	22.9	33.1	59.4	44.7	47.0
AGC	48.9	43.2	41.9	50.0	44.9	46.8	43.2	42.8	35.3	33.8	41.5	40.1	N/A	N/A
SDCN	37.5	50.8	20.0	62.3	22.4	50.3	25.1	45.6	19.7	53.3	33.0	55.7	32.1	52.8
DAEGC	56.8	33.5	47.6	36.4	53.6	37.5	39.0	43.3	19.3	58.0	39.4	49.1	N/A	N/A
k-m(DGI)	28.0	64.0	17.5	73.7	82.9	9.6	61.9	22.8	51.5	35.1	35.1	57.8	38.6	51.2
NOCD	14.7	78.3	6.8	84.4	21.7	69.6	26.4	59.0	13.7	70.1	20.9	72.2	25.7	65.5
DiffPool	26.1	66.3	26.0	63.4	32.9	56.8	35.6	30.4	26.5	46.8	33.6	59.3	N/A	N/A
MinCutPool	23.3	70.3	14.1	78.9	29.6	63.1	N/C	N/C	N/C	N/C	22.7	70.5	27.8	64.3
Ortho	28.0	65.6	18.4	74.5	57.8	32.9	N/C	N/C	N/C	N/C	27.8	65.7	33.0	59.5
DMoN	12.2	76.5	5.1	79.3	17.7	65.4	18.0	59.0	12.7	70.1	17.5	72.4	**18.8**	65.8
DGCluster($\lambda = 0.0$)	**8.4**	**80.7**	5.5	87.2	20.6	72.5	17.4	61.3	7.5	70.9	**14.6**	**74.3**	20.6	**67.5**
DGCluster($\lambda = 0.2$)	9.7	80.8	**4.1**	**87.4**	20.4	72.8	**17.7**	**61.5**	**8.6**	71.6	15.3	74.2	22.3	67.3
DGCluster($\lambda = 0.8$)	14.5	78.6	6.5	86.3	24.6	71.2	27.3	60.3	12.4	71.6	18.1	73.3	21.3	66.0

Table 2: Performance across datasets evaluated using graph conductance C and graph modularity Q, with three λ settings (0, 0.2, 0.8) from our method. The best and second-best methods are highlighted in **bold** and underlined for each dataset-metric pair. We fixed the optimal λ value per dataset in our method (i.e., 0.0 or 0.2) during comparison, and our method demonstrates the best or comparable performance to the baselines. Unavailable results and non-convergence are labeled as N/A and N/C.

	Cora		Citeseer		Pubmed		Amazon PC		Amazon Photo		Coauthor CS		Coauthor PHY	
method	NMI $\uparrow$	F1 $\uparrow$	NMI $\uparrow$	F1 $\uparrow$	NMI $\uparrow$	F1 $\uparrow$	NMI $\uparrow$	F1 $\uparrow$	NMI $\uparrow$	F1 $\uparrow$	NMI $\uparrow$	F1 $\uparrow$	NMI $\uparrow$	F1 $\uparrow$
k-m(feat)	18.5	27.0	24.5	29.2	19.4	24.4	21.1	19.2	28.8	19.5	35.7	39.4	30.6	42.9
SBM	36.2	30.2	15.3	19.1	16.4	16.7	48.4	34.6	59.3	47.4	58.0	47.7	45.4	30.4
k-m(DW)	24.3	24.8	27.6	24.8	22.9	17.2	38.2	22.7	49.4	33.8	72.7	61.2	43.5	24.3
AGC	34.1	28.9	25.5	27.5	18.2	18.4	51.3	35.3	59.0	44.2	43.3	31.9	N/A	N/A
SDCN	27.9	29.9	31.4	41.9	19.5	29.9	24.9	45.2	41.7	45.1	59.3	54.7	50.4	39.9
DAEGC	8.3	13.6	4.3	18.0	4.4	11.6	42.5	37.3	47.6	45.0	36.3	32.4	N/A	N/A
k-m(DGI)	52.7	40.1	40.4	39.4	22.0	26.4	22.6	15.0	33.4	23.6	64.6	51.9	51.0	30.6
NOCD	46.3	36.7	20.0	24.1	25.5	20.8	44.8	37.8	62.3	60.2	70.5	56.4	51.9	28.7
DiffPool	32.9	34.4	20.0	23.5	20.2	26.3	22.1	38.3	35.9	41.8	41.6	34.4	N/A	N/A
MinCut	35.8	25.0	25.9	20.1	25.4	15.8	N/C	N/C	N/C	N/C	64.6	47.8	48.3	24.9
Ortho	38.4	26.6	26.1	20.5	20.3	13.9	N/C	N/C	N/C	N/C	64.6	46.1	44.7	23.7
DMoN	48.8	48.8	33.7	**43.2**	29.8	33.9	49.3	45.4	63.3	61.0	69.1	59.8	56.7	42.4
DGCluster($\lambda = 0.0$)	49.9	42.0	26.3	20.0	24.9	29.0	51.0	47.5	68.0	64.3	72.0	72.3	54.7	40.7
DGCluster($\lambda = 0.2$)	53.0	43.5	30.3	22.2	27.6	30.1	53.8	49.5	73.0	70.7	76.1	77.3	59.0	41.9
DGCluster($\lambda = 0.8$)	**62.1**	**54.5**	**41.0**	32.2	**32.6**	**34.6**	**60.4**	**52.2**	**77.3**	**75.9**	**82.1**	**83.5**	**65.7**	**49.2**

Table 3: Performance across datasets evaluated using NMI and F1-score metrics, with three λ settings (0, 0.2, 0.8) from our method. The best and second-best methods are highlighted in **bold** and underlined for each dataset-metric pair. We fixed the optimal λ value per dataset in our method (i.e., 0.8) during comparison, and our method demonstrates the best or comparable performance across most datasets. Unavailable results and non-convergence are labeled as N/A and N/C.

	C				Q				NMI				F1			
	GCN	GAT	GIN	SAGE	GCN	GAT	GIN	SAGE	GCN	GAT	GIN	SAGE	GCN	GAT	GIN	SAGE
Cora	9.7	10.2	7.3	8.5	80.8	80.8	80.2	78.4	53.0	51.9	53.2	53.4	43.5	41.3	46.7	**51.9**
CiteSeer	4.1	4.1	4.5	4.8	87.4	87.4	87.3	84.8	30.3	29.4	29.5	32.5	22.2	21.9	21.9	**34.0**
PubMed	20.4	19.1	15.6	**10.4**	72.8	73.8	75.0	72.4	27.6	26.1	25.7	27.9	30.1	26.8	29.2	**40.5**
Amazon PC	17.7	23.4	21.3	**11.1**	61.5	61.4	54.3	55.1	53.8	52.9	35.4	45.0	49.5	45.3	42.3	52.2
Amazon Photo	8.6	8.5	6.9	5.2	71.6	72.1	69.9	63.8	73.0	72.1	61.8	60.6	70.7	69.4	60.6	58.2
Coauthor CS	15.3	15.7	14.7	13.6	74.2	74.0	73.8	73.0	76.1	75.7	71.8	72.4	77.3	79.0	72.6	74.4
Coauthor PHYSICS	22.3	20.3	18.5	**14.5**	67.3	67.1	66.7	66.0	59.0	59.2	59.7	**64.0**	41.9	43.4	48.7	**55.9**
AVERAGE	14.0	14.5	12.7	**9.7**	73.6	**73.8**	72.5	70.5	**53.3**	52.5	48.2	50.8	47.9	46.7	46.0	**52.4**

Table 4: Performance of our method with various GNN base models assessed across different metrics for all datasets, along with the calculation of average performance across datasets. Notably, the performance remains consistent across different variants of GNN models, demonstrating the absence of a clear winner. This shows the robustness of our model in adapting to diverse GNN architectures. For each dataset, we emphasize performance variations: a performance increase of more than 5 is highlighted in **bold**, while a decrease is marked with an underline, both in comparison to GCN. Similarly, in the average row, we identify the best-performing GNN base model in **bold** and the worst-performing with an underline, considering all metrics across datasets.

Auxiliary Information Effect

To check the effect of our auxiliary information, we vary the λ hyperparameter in our additional experiments. Figure **??** illustrates the impact of the hyperparameter on different metrics and datasets. The main goal of the λ parameter is to find the best underlying partition leveraging both graph structure and auxiliary information. We can see modularity Q has a consistent value over increased λ, while NMI values are increasing with a good amount. This shows the power of our auxiliary objective which ensures flexibility to the user aiming to optimize their desired metric. Additionally, the standard deviation of our results is small showing the stability of our algorithm.

	CORA		CITESEER		PUBMED	
DGCLUSTER	$Q\uparrow$	NMI $\uparrow$	$Q\uparrow$	NMI $\uparrow$	$Q\uparrow$	NMI $\uparrow$
$\alpha = 0.0$	80.8	43.5	87.4	22.2	72.8	30.1
$\alpha = 0.5$	80.8	40.3	87.3	20.3	72.1	28.1
$\alpha = 1.0$	80.8	38.5	87.4	19.5	71.7	26.3

Table 5: Performance of our method with varying α regularization parameter. Additional regularization has a negligible effect while avoiding potential trivial clustering.

Additional Regularization In addition to the primary and the auxiliary information objective terms, we propose another regularization term, the squared average node similarity, $\alpha(\frac{1}{n^2}\sum_{ij}\langle X_i X_j\rangle)^2 = \alpha\|\bar{X}\|_2^4$, where $\bar{X}$ is the mean of the node embedding vectors and α is a tunable parameter. This regularizer can be used to avoid the formation of a trivial clustering where all the nodes form a single cluster. Table **??** shows that adding this regularizer have a negligible effect on our main objectives while avoiding potential trivial clustering.

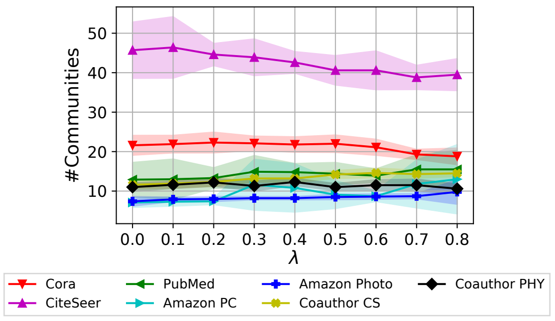

Figure 2: The number of communities varying λ hyperparameter for different datasets. We observe that the number of communities found is similar across different λ values except in some cases.

Number of Communities

Our method does not assume the number of communities in the data, which is one of its main strengths. The number of communities in real-world data is generally unknown, and it is nontrivial to estimate it. To share some insight, we provide the number of communities our method finds across different datasets and λ values. Figure **??** demonstrates the number of communities varies for different datasets, and the results are consistent across different λ values.

Conclusion

In this paper, we have studied the problem of graph clustering via maximizing modularity. The existing techniques often require the number of clusters beforehand or they fail to capitalize on the potential benefits of associated node attributes and availability of supplementary information. To address these, we have introduced DGCLUSTER, a novel neural framework that works well without a predefined number of clusters. Moreover, our framework uses graph neural networks (GNNs) to leverage the node attributes and additional information present within the graph. The computational complexity of DGCLUSTER scales linearly with the graph size. Through extensive experimentation across seven real-world datasets of varying sizes and employing multiple distinct cluster quality evaluation metrics, we have showcased the superior performance of DGCLUSTER: it consistently has outperformed state-of-the-art approaches across most of the settings.

Acknowledgments

This material is based upon work partially supported by the National Science Foundation under grant no. 2229876 and in part by funds provided by the National Science Foundation, Department of Homeland Security, and IBM.