Unsupervised Graph Clustering with Deep Structural Entropy

Jingyun Zhang
Beihang University
Beijing, China
zhangjingyun@buaa.edu.cn

Hao Peng*
Beihang University
Beijing, China
penghao@buaa.edu.cn

Li Sun
NCEPU
Beijing, China
ccesunli@ncepu.edu.cn

Guanlin Wu
School of Systems Engineering, TUDT
Changsha, China
wuguanlin16@nudt.edu.cn

Chunyang Liu
Didi Chuxing Technology Co., Ltd.
Beijing, China
liuchunyang@didiglobal.com

Zhengtao Yu
KUST
Yunnan, China
yuzt@kust.edu.cn

ABSTRACT

Research on Graph Structure Learning (GSL) provides key insights for graph-based clustering, yet current methods like Graph Neural Networks (GNNs), Graph Attention Networks (GATs), and contrastive learning often rely heavily on the original graph structure. Their performance deteriorates when the original graph's adjacency matrix is too sparse or contains noisy edges unrelated to clustering. Moreover, these methods depend on learning node embeddings and using traditional techniques like k-means to form clusters, which may not fully capture the underlying graph structure between nodes. To address these limitations, this paper introduces **DeSE**, a novel unsupervised graph clustering framework incorporating **De**ep **S**tructural **E**ntropy. It enhances the original graph with quantified structural information and deep neural networks to form clusters. Specifically, we first propose a method for calculating structural entropy with soft assignment, which quantifies structure in a differentiable form. Next, we design a Structural Learning layer (SLL) to generate an attributed graph from the original feature data, serving as a target to enhance and optimize the original structural graph, thereby mitigating the issue of sparse connections between graph nodes. Finally, our clustering assignment method (ASS), based on GNNs, learns node embeddings and a soft assignment matrix to cluster on the enhanced graph. The ASS layer can be stacked to meet downstream task requirements, minimizing structural entropy for stable clustering and maximizing node consistency with edge-based cross-entropy loss. Extensive comparative experiments are conducted on four benchmark datasets against eight representative unsupervised graph clustering baselines, demonstrating the superiority of the DeSE in both effectiveness and interpretability.

KEYWORDS

Unsupervised clustering, Graph structure learning, Structural entropy

*Corresponding author.

KDD '25, August 3-7, 2025, Toronto, Canada.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Jingyun Zhang, Hao Peng, Li Sun, Guanlin Wu, Chunyang Liu, and Zhengtao Yu. 2025. Unsupervised Graph Clustering with Deep Structural Entropy. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2025 (KDD '25), August 3-7, 2025, Toronto, Canada.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Graph structure learning (GSL) has a wide range of applications in recommender systems [29], community detection [45], interest discovery [46], web topic mining [43], graph clustering [10, 14], dimensionality reduction [20], etc. It integrates with downstream tasks to refine the graph topology and generate node classifications, enabling the learning of robust semantic embeddings and structural information, thereby improving the performance of various applications. Unsupervised graph clustering typically employs contrastive loss to learn an appropriate graph structure embedding, enhancing the representational similarity of nodes within clusters while avoiding dependence on labeled data.

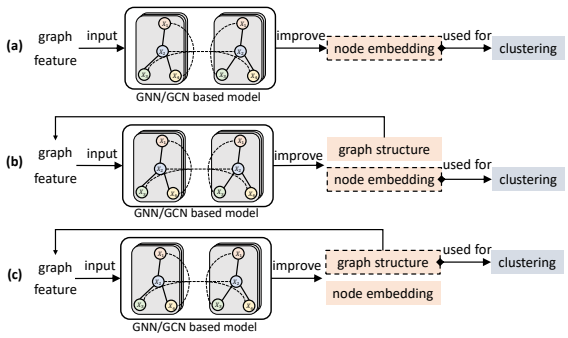


Figure 1: Concept maps of three type models. ((a) and (b) are existing models, (c) is our DeSE)

Early unsupervised graph clustering methods rely heavily on the original graph structure and focus solely on optimizing it within the model. Examples include hierarchical graph learning methods [15, 34, 39], pooling techniques [2], and structure-based embedding learning approaches [19, 25]. The primary goal of these methods is to learn better node embeddings by minimizing the distance between neighboring or structurally similar nodes, as shown in Figure 1(a). However, nodes with similar attributes may not always have direct connections in practice. For instance, papers in the same category often lack direct or indirect citation links in citation networks. This dependence on the original graph, which is typically

a sparse adjacency matrix, significantly limits the performance of the models.

To address these issues, existing methods optimize the original graph structure through approaches such as graph simplification [5, 8] and latent structure learning [7, 16, 33]. Specifically, these approaches use graph contrastive techniques to extract structural knowledge [6, 18, 35], or graph autoencoders to simultaneously learn representations and perform clustering tasks [21, 44], in order to mitigate feature drift. However, these methods still rely on the learned embeddings to form clusters, as shown in Figure 1(b). The most common approach is the classic K-Means algorithm, which requires prior knowledge of the number of clusters. We argue that such a model relies on both the quality of representation learning and the configuration of the clustering algorithm. Moreover, models that first learn embeddings and then perform clustering do not directly capture the essential relationship between node features and adaptive clusters during model convergence. Additionally, although these methods focus on unsupervised graph clustering and produce structured clusters, few models quantitatively represent the graph structure, leading to poor interpretability. While [25] proposes optimizing the node assignment matrix using modularity, this measure captures the difference between actual intra-cluster edges and expected edges, where nodes with higher degrees are more likely to be connected. This approach is not applicable in all scenarios. In tasks like citation networks and social event networks, each cluster often contains a central node closely connected to other nodes. Still, these central nodes belong to different categories and have limited connections with one another.

In this work, we propose **DeSE**, a novel unsupervised graph clustering framework with **De**ep **S**tructural **E**ntropy, which enhances the original graph using quantifiable structural information and deep neural networks to improve clustering performance and interpretability. First, in terms of structural quantification, we introduce structural information theory and propose a new method for calculating soft assignment structural entropy in the context of the graph clustering task. We transform structural entropy into a continuous and differentiable form by utilizing a probability matrix that assigns cluster nodes. This allows for retaining more information about boundary nodes during the embedding aggregation process rather than discarding low-probability nodes outright. Second, we design a Structure Learning Layer (SLL) to enhance the original graph structure. By constructing a K-nearest neighbor graph in the node feature mapping space, we create an attribute graph to address the sparsity and missing interactions between graph nodes. This attribute graph is continuously optimized and refined during the training process. Third, we propose a cluster assignment method (ASS) based on Graph Neural Networks, directly in the enhanced graph rather than using embeddings, as shown in Figure 1(c). ASS employs two convolutional layers: one for learning the embeddings of the current layer's nodes and the other for learning the soft assignment matrix. These are then aggregated to obtain embeddings for higher-level communities and update the graph structure at the upper levels. Finally, the model is optimized by minimizing the structural entropy of the assignments to stabilize the cluster structure and by using an edge-based cross-entropy loss to maximize the consistency between connected nodes, thereby achieving unsupervised graph clustering.

We conduct extensive experiments on four datasets, Cora, Citeseer, Computer, and Photo, to demonstrate the effectiveness of DeSE. First, the overall experimental results indicate that DeSE demonstrates superior overall performance compared to the eight baseline models. Second, a series of ablation experiments analyze the SLL, GNN layer for features, and structure entropy loss DeSE. Thirdly, the experiment for hyperparameters also illustrates the high performance and stability of DeSE. The main contributions of this work are summarized as follows.

- A novel unsupervised graph clustering framework with Deep Structural Entropy is proposed with high effectiveness.
- A structure learning layer (SLL) and a cluster assignment method (ASS) based on Graph Neural Networks are designed for enhancing structure and graph clustering.
- A new optimization method that minimizes the structural entropy of the assignments and maximizes the consistency between connected nodes.
- A series of comparative analysis experiments show that DeSE achieves higher graph clustering effectiveness and strong interpretability.

2 RELATED WORK

2.1 Unsupervised Graph Clustering

Unsupervised graph clustering has evolved significantly over the past decades, from traditional methods like spectral clustering [27] and modularity-based approaches [22] to the more sophisticated deep learning models used today. Early methods primarily focused on leveraging graph structure alone, which limited their performance in dealing with complex, feature-rich data. Spectral clustering and its variants, which use eigenvalue decomposition of graph Laplacians, have been widely used due to their theoretical simplicity and effectiveness [30]. However, these approaches scale poorly to large graphs and are noise-sensitive, especially when the graph structure is sparse or incomplete. With the advent of deep learning, there has been a shift towards models that integrate node features and graph structure. Techniques such as Graph Neural Networks (GNNs), particularly Graph Convolutional Networks (GCNs) [11], and Graph Autoencoders (GAEs) [12], have gained traction for unsupervised graph clustering tasks. These models learn node embeddings that preserve local and global structures, facilitating better clustering results. Furthermore, some approaches have started to explore the role of structure in enhancing clustering [15, 34]. These methods aim to quantify and improve the quality of graph structures, particularly in the presence of noise, leading to more stable clustering results.

2.2 Graph Structure Learning

Graph Structure Learning (GSL) has gained increasing attention in recent years as researchers seek to optimize graph structures for downstream tasks like clustering and node classification. Unlike traditional methods that rely on predefined graphs, GSL techniques dynamically learn or refine the graph structure based on node features and interactions. This approach has shown to be particularly effective in handling noisy, incomplete, or poorly defined graphs. Adaptive graph learning methods [5, 9] focus on pruning noisy edges or adding informative ones to improve graph quality. These models typically apply sparsity constraints or similarity metrics to update the graph structure during training, resulting in a

cleaner and more informative graph representation. Joint learning approaches [9, 40] simultaneously learn the graph structure and node embeddings in an end-to-end manner. These methods are particularly powerful because they allow structure optimization based on the specific task. A key challenge in GSL is balancing structural refinement with maintaining meaningful graph relationships. These techniques show promise in handling noisy graphs and enhancing overall task performance, especially in unsupervised or semi-supervised settings.

2.3 Structural Information Theory

The Structural Information Theory decoding network's ability to capture the structure's essence has been validated in many applications. Introducing structural entropy in neural networks captures the underlying connectivity graph and reduces random interference [28]. The hierarchical nature of the structure entropy encoding tree provides new methods for hierarchical structure pooling in graph neural network [32], unsupervised image segmentation [41], dimension estimation [38], state abstraction [42] in reinforcement learning, social bot detection [23, 36], and unsupervised social event detection [4]. Additionally, reconstructing the graph structure on the hierarchical encoding tree suppresses edge noise and enhances the learning ability of the graph structure [47, 48]. Furthermore, modifying the network structure based on minimizing structural entropy achieves maximum deception of community structure [17]. Similarly, the anchor view, guided by the principle of minimizing structural entropy, improves the performance of graph contrastive learning [31]. Based on the homogeneous graph structure entropy, the study of multi-relational graph structure entropy [3] further extends the structural information theory, making it suitable for more complex scenarios.

3 PRELIMINARY

*Definition 3.1 (**Unsupervised Graph Clustering**).* The unsupervised graph clustering task aims to cluster nodes based solely on the provided interaction and feature information. From a data perspective, the input consists of an undirected homogeneous graph with node features, represented as $G = (V, \mathcal{E}, X)$, where V is a set of N nodes, $\mathcal{E}$ is a set of M edges, and $X \in R^{N\times f}$ is the node feature matrix with dimension f. The edge relationships between nodes in G are represented by a symmetric adjacency matrix $A_g \in \{0,1\}^{N\times N}$, where each element denotes the edge weight between nodes. The objective of the unsupervised graph clustering task is to learn a node assignment matrix $S \in \{0,1\}^{N\times c}$, which reflects the community memberships of nodes, where c is the number of clusters. In our model, c does not need to be specified in advance.

*Definition 3.2 (**Structural Entropy**).* Structural information theory [13] is originally proposed for measuring the structural information contained within a graph. Specifically, this theory aims to calculate the structural entropy of the homogeneous graph $G = (V, \mathcal{E})$, which reflects its uncertainty when undergoing hierarchical division. The structural information of the homogeneous graph G determined by the encoding tree $\mathcal{T}$ is defined as:

$$H^{\mathcal{T}}(G) = -\sum_{\alpha\in\mathcal{T},\alpha\neq\lambda} \frac{g_\alpha}{vol(G)} log\frac{vol(\alpha)}{vol(\alpha^-)}, \tag{1}$$

where $vol(G)$ is the sum of the degrees of all nodes in the graph G. Each vertice in the encoding tree $\mathcal{T}$ corresponds to a node subset T_α in the graph G. $vol(\alpha)$ is the volume of T_α and is the sum of the degrees of all nodes in the subset T_α. α^- is the parent vertice of vertice α in the encoding tree. g_α is the sum of weights of all edges from node subset T_α to node subset V/T_α, which can be understood as the total weight of the edges from the nodes outside the node subset T_α to the nodes inside T_α, or the total weight of the cut edges. $\frac{g_\alpha}{vol(G)}$ represents the probability that the random walk enters T_α. The structural entropy $H(G)$ of graph G is the minimum $H^{\mathcal{T}}(G)$. Let $\mathcal{T}_k$ be encoding trees whose height is not greater than k, then the k-dimensional structural entropy of G is defined as $H_k(G) = min\{H^{\mathcal{T}_k}(G)\}$.

4 METHODOLOGY

This section elaborates on the unsupervised graph clustering framework DeSE with deep structural entropy. As shown in Figure 2, DeSE consists of three key modules: Structural Quantification, Structural Learning Layer, and Clustering Assignment Layer. Specifically, **Structural Quantification** (Section 4.1) introduces a soft assignment structural entropy, quantifying the structural information and transforming discrete clustering into a continuous and differentiable objective. **Structural Learning Layer (SLL)** (Section 4.2) learns a K-nearest neighbor attribute graph in the feature mapping space to enhance the original graph structure. **Clustering Assignment Layer (ASS)** (Section 4.3), based on GNN, simultaneously learns node embeddings and a soft assignment matrix in the enhanced graph, updating new cluster embeddings and the cluster-enhanced graph. **Optimization** (Section 4.4) integrates all modules, optimizing the learning process using structural entropy loss between nodes and clusters and cross-entropy loss between node embeddings.

4.1 Structural Quantification

Structural information theory offers significant advantages in learning hierarchical structures and clusters of graph nodes. It quantifies the uncertainty in graph structures and represents them in a mathematically computable form. Although prior research has extended structural entropy [3, 23] and its optimization methods from simple homogeneous graphs to multi-relational graphs and hypergraphs, structural entropy is still calculated in a discrete manner. This limitation restricts current optimization methods to operations like the merge operator, where nodes are greedily merged in pairs. However, in unsupervised graph clustering tasks, we aim for structural entropy to not only divide nodes into clusters but also provide trainable feedback to enhance the graph structure. This highlights the limitations of traditional structural entropy.

To address this issue, we transform the original binary "belong/not belong" relationship between nodes and clusters (represented as discrete values of 0 or 1 in the assignment matrix) into a probabilistic relationship. A node is no longer exclusively assigned to a single cluster; instead, it can belong to multiple clusters with varying probabilities. This approach aligns with real-world scenarios, such as interdisciplinary papers in citation networks. Although these papers have a primary category, they also contribute to and are associated with other relevant categories, which is valuable information in embedding and structure learning. This probabilistic cluster assignment is also known as a "soft assignment."

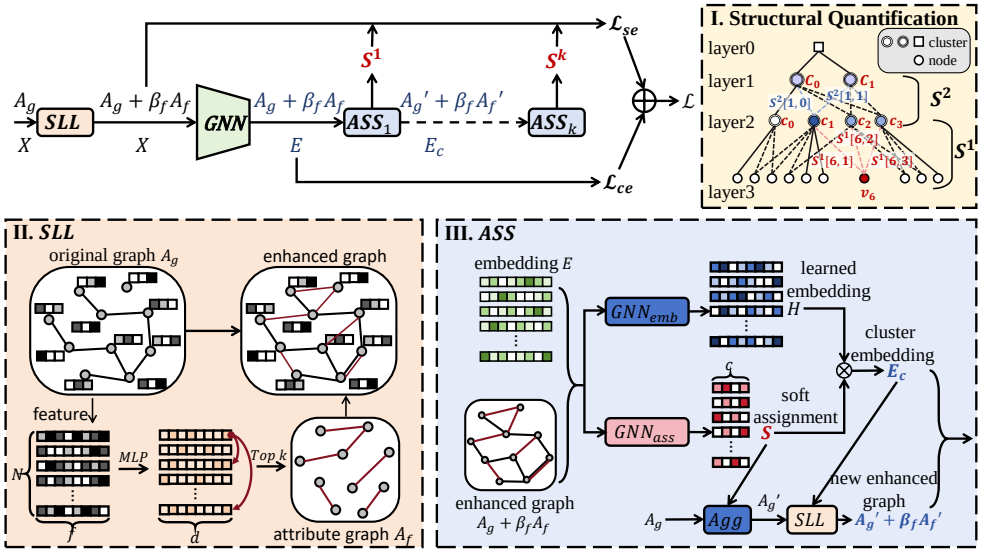


Figure 2: The overall framework of DeSE.

Soft Assignment SE. The traditional definition of structural entropy is presented in Section 3. Structural information theory quantifies the uncertainty in a graph's structure based on the random walk of nodes through edges. When the lower-level vertices belong to the parent vertex according to the assignment matrix S^k at layer k, we first express Eq. 1 as the sum of node entropies at each layer and introduce the concept of a direct assignment matrix:

$$H_{sa}^{\mathcal{T}}(G) = \sum_{k=1}^{h} H_{sa}(G;k), \tag{2}$$

$$C^k = S^h \cdot S^{h-1} \cdot \ldots \cdot S^{k+1}, \tag{3}$$

where $H_{sa}(G;k)$ in Eq. 2 denotes the structural entropy at layer k with N_k vertices, while the total height of the encoding tree is h. The $S^k \in R^{N_k \times N_{k-1}}$ represents the assignment matrix between the vertices of layer k and those of layer $k-1$. As computed in Eq. 3, $C^k \in R^{N \times N_k}$ is the direct assignment matrix between the leaf vertices (i.e., the nodes in the graph) and the vertices of layer k, representing the probability that each node belongs to a cluster at layer k. Additionally, we redefine the calculation and representation of cut edges and volume as follows:

$$vol^k[i] = D(C^k)_i, \tag{4}$$

$$g_i^k = vol^k[i] - vol_{in}^k[i] = vol^k[i] - (C^k)_i{}^{\top} W (C^k)_i, \tag{5}$$

where the volume $vol^k[i]$ of vertex i at layer k is the sum of the assignment probabilities over all node degrees, as expressed in Eq. 4. And $D \in R^N$ is the degree vector for all nodes, which can be obtained through the weight matrix of edges $W \in R^{N \times N}$ that $D = \mathbf{1_N}^{\top} \cdot W$ ($\mathbf{1_N}^{\top}$ is a length-N vector consisting entirely of ones and the calculation of weight matrix W will be detailed in Section 4.2). The subscript i on C^k refers to the i-th column vector of the matrix, which represents the direct clustering probability of N nodes in the graph to the i-th cluster at layer k. And the i-th vertex at layer k represents the i-th cluster at layer k. The term g_i^k represents the cut value of the i-th vertex at layer k, which is calculated as the difference between the volume $vol^k[i]$ of vertex i and its internal volume $vol_{in}^k[i]$. The internal volume $vol_{in}^k[i]$ is expressed as the sum of the weighted probabilities of all edges, where the probability refers to the likelihood that the two nodes connected by an edge belong to the same cluster i at layer k. Thus, the computation of structural entropy is modified as follows:

$$\begin{aligned} H_{sa}(G;k) &= -\sum_{i=1}^{N_k} \frac{g_i^k}{vol^0} log \frac{vol^k[i]}{\sum_{j=1}^{N_{k-1}} vol^{k-1}[j] \cdot S^k[i,j]} \\ &= -\sum_{i=1}^{N_k} \frac{vol^k[i] - (C^k)_i{}^{\top} W (C^k)_i}{vol^0} log \frac{vol^k[i]}{vol^{k-1}(S^k)^{\top}{}_i} \\ &= -\sum_{i=1}^{N_k} \frac{D(C^k)_i - (C^k)_i{}^{\top} W (C^k)_i}{DC^0} log \frac{D(C^k)_i}{DC^{k-1}(S^k)^{\top}{}_i}, \end{aligned} \tag{6}$$

where the original volume associated with a single parent vertex is replaced by the probabilistic sum of the volumes of all parent vertices $vol^{k-1}(S^k)^{\top}{}_i$ in the soft assignment approach. And $vol^{k-1} = [vol^{k-1}[1], ..., vol^{k-1}[N_{k-1}]]$ is the vector representation of the volume of all N_k vertices at layer k, which can be further simplified to DC^{k-1} by Eq. 4

4.2 Structural Learning Layer (SLL)

In graph clustering tasks, the input typically consists of node features X and an adjacency matrix A_g. The general approach is to train the node features based on the original structure to generate embeddings. However, the connections in A_g do not always align perfectly with clustering objectives. For instance, in citation networks, papers on similar topics may not be directly linked (e.g., papers on neural networks but focusing on different application areas), or papers that are linked may not belong to the same topic (e.g., interdisciplinary or multi-methodology papers). Similarly, in product co-purchase networks, items bought together may be complementary rather than similar (e.g., desktop computers and monitors or cameras and lenses). Such mismatches in the original graph structure A_g will lead to information loss during the aggregation process, which can hinder the effectiveness of graph clustering.

The SLL aims to enhance the original graph structure by leveraging the available feature information X and dynamically optimizing and updating the graph during training. Since the original features are often high-dimensional and sparse binary vectors, we first map the node features into a lower-dimensional dense space using a multilayer perceptron (MLP), as shown in Figure 2 II. Based on these embeddings, we apply the K-nearest neighbors algorithm (KNN) to select the top K neighbors for each node and create edges with a weight of 1. This process constructs an attribute graph, mathematically represented as:

$$A_f = KNN(MLP(X;\Theta_f);K), \tag{7}$$

where MLP is a multilayer perceptron with an input size of f and both hidden and output layers of size d, while $\Theta_f \in R^{f\times d}$ represents the parameters of the MLP. KNN is the K-nearest neighbors operation, and each row of the obtained adjacency matrix $A_f \in \{0,1\}^{N\times N}$ indicates the neighbor selection for each node. Since KNN selects neighbors by ranking the distances between nodes, the neighbors may be unidirectional. That is if the node v_i is among the top K neighbors of node v_j, node v_j may not necessarily be among the top K neighbors of node v_i. Given the different implications of unidirectional versus bidirectional neighbor selection, we adjust the adjacency matrix of the attribute graph as follows:

$$A_f = (A_f + A_f{}^\intercal)/2. \tag{8}$$

This ensures that the attribute graph's adjacency matrix becomes symmetric while still partially retaining the unidirectional and bidirectional neighbor selections. Finally, we combine the original graph adjacency matrix with the attribute graph adjacency matrix to obtain an enhanced graph:

$$W = A_g + \beta_f A_f, \tag{9}$$

where β_f is a hyperparameter controlling the weight of the attribute graph in the fusion, and $W \in R^{N\times N}$, is the new adjacency matrix with edge weights used for soft assignment SE in Section 4.1 and subsequent calculations.

4.3 Clustering Assignment Layer (ASS)

The clustering assignment layer utilizes the initial embeddings and the adjacency matrix to learn the soft assignments and embeddings of nodes while updating the graph structure and cluster embeddings after aggregation. It consists of three components: Embedding Learner GNN_{emb}, Soft Assignment Learner GNN_{ass}, and Aggregator Agg, as shown in Figure 2 III.

Embedding Learner. The embedding learner is based on a GNN architecture, mathematically represented as follows:

$$H = GNN_{emb}(E,W;\Theta_1) = ReLU(mean(WE\Theta_1)), \tag{10}$$

where GNN_{emb} applies a linear transformation to the initial embedding E, mapping it to the embedding space. It then aggregates the average embeddings of connected nodes to generate new node embeddings, followed by an activation function. The learnable parameters of the linear transformation are denoted by $\Theta_1 \in R^{d\times d}$.

Soft Assignment Learner. The soft assignment learner extends the GNN architecture with an attention mechanism, mathematically represented as follows:

$$\begin{aligned} S &= GNN_{ass}(E,W;\Theta_2) = ReLU((\Gamma \circ W)E\Theta_2), \\ \Gamma_{i,j} &= \frac{LeakyReLU((e_i||e_j)\Theta_3))}{\sum_{j=1}^{N} LeakyReLU((e_i||e_j)\Theta_3))}, \end{aligned} \tag{11}$$

where GNN_{ass} performs a linear transformation on the initial embeddings E into the cluster space (i.e., with dimensions equal to the number of clusters). It then computes the attention matrix Γ for each edge to serve as aggregation weights, performing non-averaged embedding aggregation to obtain cluster embeddings. The learnable parameters of the linear transformation in GNN_{ass} are denoted by $\Theta_2 \in R^{d\times c}$. The computation of attention involves linearly transforming the concatenated embeddings of the nodes at each end of an edge into a 1-dimensional space (i.e., the weight space), followed by activation and normalization. e_i and e_j represent the embedding of node v_i and node v_j. The learnable parameters for the linear transformation in the attention mechanism are denoted by $\Theta_3 \in R^{2d\times 1}$.

Aggregator. The aim of Agg is to update the embedding and adjacency matrix of clusters. We use the probability sum of node embeddings for cluster embeddings, denoted as $E_c = S^\intercal H \in R^{c\times d}$. The new adjacency matrix is combined from the attribute graph adjacency matrix and the structural graph adjacency matrix, similar to the structural learning method described in Section 4.2 Eq. 9. The detailed computation process is as follows:

$$A_g{}' = S^\intercal A_g S, \quad A_f{}' = KNN(MLP(E_c;\Theta_c);K), \tag{12}$$

where $\Theta_c \in R^{d\times d}$ represents the learnable parameters of the MLP for clusters and the new weighted adjacency matrix is $W' = A_g{}' + \beta_f A_f{}'$.

4.4 Optimization

The entire process of graph clustering in DeSE is illustrated in Appendix B Algorithm 1. Initially, the original graph is enhanced as detailed in Section 4.2, followed by a round of GNN propagation on the new weighted adjacency matrix W, transforming the sparse and high-dimensional feature vectors X into the initial node embeddings E. Next, several ASS (Section 4.3) modules are used to learn soft assignment matrices $\{S^k\}$ at different layers. We employ soft assignment structural entropy (SE loss) and negative sampling cross-entropy loss (CE loss) to optimize the graph clustering task. Let the set of positive and negative edges be denoted as $\mathcal{E}'$, with an equal number of positive and negative edges. The CE loss is then calculated as follows:

$$\begin{aligned} p_{i,j} &= Sigmoid(2 - ||e_i - e_j||_2), \quad l_{i,j} = \begin{cases} 1, & W_{i,j} \neq 0 \\ 0, & else \end{cases}, \\ \mathcal{L}_{ce} &= -\frac{1}{|W|} \sum_{(i,j)\in\mathcal{E}'} l_{i,j} log(p_{i,j}) + (1 - l_{i,j}) log(1 - p_{i,j}), \end{aligned} \tag{13}$$

where $p_{i,j}$ represents the probability of an edge existing between node v_i and node v_j, calculated based on the distance between their embeddings. Let $l_{i,j}$ denote the ground truth label indicating whether an edge actually exists between them. The final loss is composed of SE loss, as calculated in Section 4.1, and CE loss:

$$\mathcal{L} = \lambda_{se} H_{sa}^{\mathcal{T}} + \lambda_{ce} \mathcal{L}_{ce}, \tag{14}$$

Table 1: Comparison of the NMI, ARI, ACC, and F1 across different methods on four datasets. The best results are bolded, and the second-best results are underlined.

Method	Cora				Citeseer				Computer				Photo			
	NMI	**ARI**	**ACC**	**F1**	**NMI**	**ARI**	**ACC**	**F1**	**NMI**	**ARI**	**ACC**	**F1**	**NMI**	**ARI**	**ACC**	**F1**
DMoN	48.98	40.55	63.55	58.91	33.70	30.27	59.12	56.68	49.30	31.82	49.13	39.29	63.38	52.41	73.83	71.05
MinCut	40.40	29.44	54.69	53.13	28.70	23.60	49.02	46.96	48.37	26.07	38.92	34.73	57.47	43.88	66.26	64.84
DGI	53.56	49.71	70.35	67.56	38.36	31.53	53.14	42.25	22.68	12.76	29.21	24.52	30.70	14.63	39.59	37.74
SUBLIME	54.20	50.30	71.30	63.50	44.10	43.90	68.50	63.20	45.55	24.89	40.94	34.64	56.91	45.11	64.95	61.24
EGAE	54.00	47.20	72.40	50.94	41.20	43.20	67.40	42.85	45.98	34.80	49.59	40.28	59.99	50.14	70.33	67.78
CONVERT	55.57	50.58	74.07	**72.92**	41.62	42.77	68.43	62.39	48.44	34.25	51.30	42.17	63.04	55.20	74.24	69.13
AGC-DRR	18.74	14.80	40.62	31.23	40.28	45.34	68.32	**64.82**	49.31	35.93	55.72	42.61	60.43	48.20	68.44	62.63
RDGAE	55.30	53.00	73.10	48.32	43.70	**45.70**	**69.50**	43.54	42.64	37.34	56.72	35.77	48.12	33.14	51.92	42.39
DeSE	**57.96**	**55.47**	**75.22**	72.63	**44.34**	45.01	69.34	64.29	**52.10**	**45.64**	**58.78**	**43.17**	**70.13**	**62.50**	**80.55**	**76.55**
Improv.(%)	↑**4.3**%	↑**4.7**%	↑**1.6**%	↓0.4%	↑**0.5**%	↓1.5%	↓0.2%	↓0.8%	↑**5.7**%	↑**22.2**%	↑**3.6**%	↑**1.3**%	↑**10.7**%	↑**13.2**%	↑**8.5**%	↑**7.7**%

Table 2: Statistics of four datasets.

Dataset	#Node	#Edge	#Feature	#Cluster	Sparsity	Iso.
Cora	2,708	5,278	1,433	7	3.9	0
Citeseer	3,327	4,552	3,703	6	2.7	48
Computer	13,752	245,861	767	10	35.7	281
Photo	7,650	119,081	745	8	31.1	115

where λ_{se} and λ_{ce} are hyperparameters of coefficients for SE loss and CE loss, respectively.

4.5 Complexity Analysis

We analyze the time complexity of each component of DeSE. For SLL, the time complexity is mainly contributed by MLP and KNN, which are $O(Nfd)$ and $O(NlogN)$, respectively. In the ASS, the complexities of embedding learner and soft assignment learner are $O(Nd^2)$ and $O(Nd^2 + Nd)$. For loss computation, the complexity is $O(Nc)$ for SE loss and $O(2Md)$ for CE loss. Since the number of clusters c is usually small, the total time complexity of the model can be summarized as $O((d + logN + f)Nd + Md)$, which can be simplified to $O(dNlogN)$.

5 EXPERIMENTS

In this section, we conduct empirical experiments to demonstrate the effectiveness of the proposed framework DeSE. We aim to answer five research questions as follows: Q1: How effective the DeSE is for unsupervised graph clustering, and what kind of clusters are learned compared with baselines (Section 5.2)? Q2: How do the Structure Learning Layer and SE loss influence the performance of DeSE (Section 5.3)? Q3: How do key hyperparameters impact the performance of DeSE (Section 5.4)? Q4: How robust is DeSE to the number of clusters, and what kind of graph structure is learned (Section 5.5)?

5.1 Experiment Setup

Datasets. We conduct experiments on four benchmark datasets: Cora, Citeseer, Computer, and Photo. Details of datasets are summarized in Table 2 of Appendix D, supplemented with the number of independent nodes "Iso." and the average number of edges per node "Sparsity".

Baselines. For graph clustering, we mainly compare DeSE with two categories of methods, including three structure-driven models (i.e., DMoN [25], MinCut [1], and DGI [26]) and five unsupervised GSL models (i.e., SUBLIME [18], EGAE [44], CONVERT [35], AGC-DRR [8], and RDGAE [21]). Details of baselines are summarized in Appendix C. The codes for all baseline models and DeSE, along with all datasets, are publicly accessible on GitHub[1].

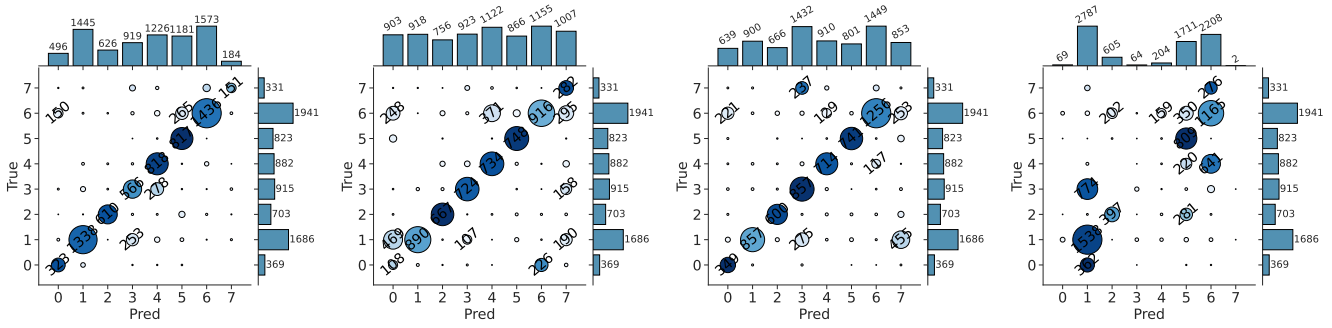
Evaluation Metrics. We evaluate the accuracy and consistency of graph clustering with four metrics. NMI (Normalized Mutual Information) evaluates how well the predicted clusters match the true clusters in terms of information shared. ARI (Adjusted Rand Index) assesses the similarity between the predicted and true cluster assignments, adjusting for random chance. ACC (Accuracy) measures the proportion of nodes correctly assigned to their true clusters. F1 Score evaluates the balance between precision and recall in cluster assignments.

5.2 Graph Clustering Performance

Table 1 reports the graph clustering results of our method DeSE in comparison with eight baseline models across four datasets. The evaluation metrics include NMI, ARI, ACC, and F1. For all methods, both the original graph structure and node features from the datasets are used as input. The baseline models are drawn from open-source implementations. As can be observed, despite the absence of labeled data, our proposed DeSE model outperforms all baselines on 12 out of the 16 evaluated metrics across the four datasets and ranks second on three of the remaining metrics. Notably, the DeSE model achieves the best performance on the NMI metric in all benchmarks. This strong performance is attributed to the novel approach of leveraging deep structural entropy to enhance graph structure learning, guiding adaptive clustering.

Additionally, we make other observations: First, the GSL (Graph Structure Learning) baselines outperform methods that directly use the raw graph structure for clustering on most datasets and metrics, demonstrating the importance of structure learning and structure enhancement. Second, while most baselines perform relatively well on NMI and ACC, they struggle to balance ARI and F1. For instance, DMoN and MinCut exhibit particularly poor ARI, while EGAE and RDGAE have notably low F1 scores. This imbalance can largely be attributed to differences in the prediction of majority and minority classes. ACC and NMI tend to focus on overall alignment, whereas F1 and ARI place greater emphasis on local precision, particularly in the handling of imbalanced classes. In contrast, our DeSE model does not exhibit such conflicts and performs well across all four metrics. Figure 3 presents the detailed correspondence between the node numbers of the predicted clusters and the true clusters on DeSE

[1]https://github.com/SELGroup/DeSE



(a) DeSE. (b) MinCut. (c) EGAE. (d) RDGAE.

Figure 3: Clusters of DeSE, EGAE, MinCut, and RDGAE on the Photo dataset. (The vertical axis represents the number of nodes contained in the true clusters, while the horizontal axis represents the number of nodes predicted by the model. Each circle in the heatmap shows the number of nodes from true cluster i predicted to belong to cluster j. The circle's size represents the node count, and its color intensity indicates the proportion of these nodes within true cluster i, with darker colors showing a higher proportion.)

Table 3: Test results corresponding without SLL module and SE loss on four datasets.

Variation	Cora				Citeseer			
	NMI	ARI	ACC	F1	NMI	ARI	ACC	F1
w/o SLL	54.21	47.72	-	-	43.00	43.56	52.63	45.69
w/o SE	35.19	24.87	47.27	39.43	24.65	19.25	42.14	34.13
DeSE	**57.96**	**55.47**	**75.22**	**72.63**	**44.34**	**45.01**	**69.34**	**64.29**
Variation	Computer				Photo			
	NMI	ARI	ACC	F1	NMI	ARI	ACC	F1
w/o SLL	51.18	33.13	55.65	43.18	70.09	62.29	79.67	73.23
w/o SE	39.19	34.21	-	-	53.05	37.50	-	-
DeSE	**52.10**	**45.64**	**58.79**	**43.17**	**70.13**	**62.50**	**80.55**	**76.55**

and three baselines. It can be observed that RDGAE predicts large clusters with relatively high accuracy but shows noticeable errors for smaller categories, such as the misprediction of the majority of nodes from cluster 0 and cluster 3 into cluster 1. In contrast, EGAE and MinCut do not provide comprehensive predictions for the larger clusters, with a significant number of nodes from cluster 1 and cluster 6 being distributed across other clusters. Our DeSE model maintains relatively focused and accurate clustering for both large and small clusters. The clusters of DeSE plots for the remaining three datasets are discussed in Appendix E.

5.3 Ablation Study

In our proposed DeSE, the Structural Learning Layer refines the graph structure, and SE loss with soft assignment is introduced for optimization. To evaluate the effectiveness of these two components, we independently disabled the SLL and SE loss (i.e., set $\beta_f = 0$ and $\lambda_{se} = 0$), resulting in the variations referred to as "w/o SLL" and "w/o SE". The corresponding results are presented in Table 3. Without the SLL, the clustering performance deteriorates more significantly on datasets with relatively sparse connections, such as Cora and Citeseer. This is especially evident in the ACC and F1 scores on Cora, where they become undefined, indicating a mismatch between the predicted and actual number of clusters. This suggests that the SLL improves the quality of the graph structure, particularly for sparse graphs. Similarly, in the absence of SE loss, clustering performance

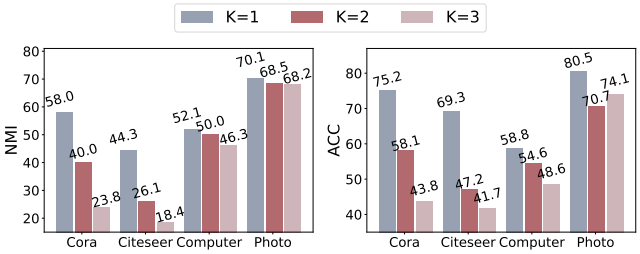


Figure 4: Sensitivity of hyperparameter K with NMI and ACC.

declines sharply across all datasets, with large and relatively dense datasets such as Computer and Photo showing undefined ACC and F1 scores due to the mismatch between the predicted and actual number of clusters. This highlights the importance of optimizing the model using quantified structural information to improve clustering performance.

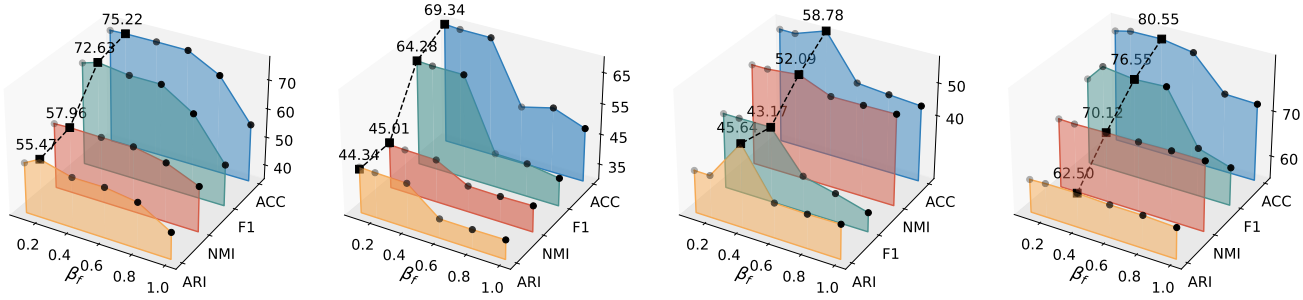
5.4 Sensitivity Analysis

In this section, we investigate the sensitivity of critical hyperparameter in DeSE, including the number of neighbors in KNN K, the weight of attribute graph β_f, and the coefficients of SE loss and CE loss λ_{se} and λ_{ce}.

Number of neighbors K. Figure 4 shows the NMI and ACC performance under different numbers of neighbors across four datasets. It can be observed that the value of $K = 1$ yields high NMI and ACC scores. As the value of K increases, both NMI and ACC show a significant decline in the Cora and Citeseer datasets, which have relatively sparse graph structures. However, on the Computer and Photo datasets, which have a higher average number of edges per node, the decrease is less pronounced. This suggests that the selection of neighbors has a significant impact on clustering results. While improving the graph structure using neighbors offers advantages for clustering, introducing too many neighbors can be detrimental, as it may introduce noise. Nodes with similar features in the embedding space do not necessarily belong to the same cluster. Next, we provide a more detailed explanation of the degree-based distributions used for selecting K and present the complete experimental results in Table 4. The first row represents the results

Table 4: Performance of DeSE on different degree-based distributions for K.

Method	Cora				Citeseer				Computer				Photo			
	NMI	ARI	ACC	F1	NMI	ARI	ACC	F1	NMI	ARI	ACC	F1	NMI	ARI	ACC	F1
$K = 1$	**57.96**	**55.47**	**75.22**	**72.63**	**44.34**	**45.01**	**69.34**	**64.29**	**52.10**	**45.64**	**58.78**	**43.17**	**70.13**	62.50	**80.55**	**76.55**
$K \sim /5$	54.72	49.57	73.33	68.70	39.98	40.38	66.39	61.62	39.89	26.54	46.16	29.51	65.79	61.42	-	-
$K \sim /10$	56.57	51.44	74.29	70.21	43.08	43.56	68.19	63.17	45.08	32.93	48.56	34.05	62.31	58.91	65.47	52.67
$K \sim sqrt$	36.20	33.74	62.81	57.55	21.47	20.32	45.77	36.76	35.27	23.40	43.20	23.12	58.76	49.59	69.68	57.92
$K \sim log$	30.04	25.37	55.31	52.10	20.73	19.90	44.72	35.93	42.96	27.47	-	-	54.49	43.75	63.07	45.62
$K \sim \hat{}$	57.88	52.67	**75.22**	71.07	42.79	43.30	68.17	63.28	45.79	32.32	56.43	30.07	65.06	**64.99**	73.79	64.30
$K \sim random$	39.17	32.21	56.28	54.39	22.25	20.90	45.00	39.45	47.86	30.00	52.07	41.47	65.75	58.68	58.31	43.25



(a) Cora. (b) Citeseer. (c) Computer. (d) Photo.

Figure 5: Sensitivity of hyperparameter β_f on four datasets with four metrics.

for $K = 1$, which are reported in our paper. The subsequent five sets of comparative experiments consider ($EPS = 1e-6$):

- $K \sim /5$: $K = ceil(\frac{degree}{5} + EPS)$,
- $K \sim /10$: $K = ceil(\frac{degree}{10} + EPS)$,
- $K \sim sqrt$: $K = ceil(\sqrt{degree} + EPS)$,
- $K \sim log$: $K = ceil(log_2(degree + 1) + EPS)$,
- $K \sim \hat{}$: $K = floor(degree^{\frac{1}{degree+1}})$.

In Table 4, a '-' indicates that the number of clusters produced by DeSE under this configuration does not match the actual number, making it impossible to compute ACC and F1. We can see that the best performance of DeSE is still at $K = 1$, as too many neighbors distort the graph while both enhancing and preserving it. Therefore, we chose $K = 1$ in our experiments. Then, to understand whether the introduced edge ($K = 1$) is noise or if it represents the missing edge in the original graph, we add a variation "$K \sim random$" where, instead of using KNN, a random edge is added and check its performance. Results in the last line of Table 4 show a significant performance drop with random edges compared with $K = 1$. Thus, while DeSE performs best when $K = 1$; we argue that this setting effectively recovers missing but meaningful edges in the original graph.

Weight of attribute graph β_f. To analyze its sensitivity, we search the value of β_f in the range of $\{0.1, 0.2, 0.4, 0.6, 0.8, 1.0\}$. As is demonstrated in Figure 5, the optimal choice of β_f varies across datasets. For example, the best performance is achieved at 0.4 for Computer and Photo, 0.2 for Cora, and 0.1 for Citeseer. However, a common trend is that an excessively large β_f leads to poor performance, particularly with F1 and ACC scores, which drop sharply when β_f exceeds 0.6. We hypothesize that this occurs because assigning too much weight to the attribute graph may interfere with the effectiveness of the original graph structure.

Table 5: Sensitivity of hyperparameters λ_{se} and λ_{ce} with NMI.

Variation	SE loss λ_{se}				CE loss λ_{ce}			
	0.01	0.05	0.2	0.5	0.1	0.5	1	5
Cora	**57.96**	56.94	55.02	54.19	54.24	56.82	57.02	**57.96**
Citeseer	**44.34**	42.30	41.41	40.88	44.01	**44.34**	44.13	44.26
Computer	40.74	44.45	**52.10**	42.26	46.05	**52.10**	47.31	43.10
Photo	**70.13**	69.95	63.90	66.06	66.08	68.67	70.13	**70.13**

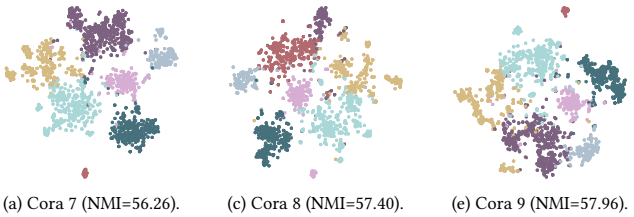
Coefficients of SE loss and CE loss λ_{se} and λ_{ce}. The NMI results of DeSE with λ_{se} in the range of $\{0.01, 0.05, 0.2, 0.5\}$ and λ_{ce} in the range of $\{0.1, 0.5, 1, 5\}$ are presented in Table 5. The optimal choice varies across different datasets, but generally, the SE loss tends to favor smaller coefficients, while the CE loss tends to favor larger coefficients. However, compared to the variant w/o SE ($\lambda_{se} = 0$) in section 5.3, it is evident that despite the small coefficient of the SE loss, it plays a significant role in improving performance.

5.5 Robustness on Clusters

To evaluate the robustness of DeSE under different cluster number settings, we set the number of clusters in GNN_{ass} to the original cluster number c, $c + 1$, and $c + 2$, respectively, and visualize the results using t-SNE. As shown in Figure 6, the first row presents the predictions of DeSE on the Cora dataset, while the second row displays the results under the ground-truth labels. It can be observed that regardless of the set number of clusters, DeSE consistently forms seven clusters, matching the ground truth, and achieves a relatively high NMI. This demonstrates the robustness of our model with respect to cluster numbers. Additionally, cluster construction is not entirely dependent on embedding learning. The ground-truth distribution shows that distant nodes can belong to the same cluster. DeSE is able to capture such nodes to some extent. For instance, in Figure 6(e), a small group of yellow nodes in the lower left is closer to the purple cluster in terms of embedding distribution, but DeSE

Table 6: Robustness of cluster numbers on baselines.

Method	Cluster	Cora				Citeseer				Computer				Photo			
		NMI	ARI	ACC	F1	NMI	ARI	ACC	F1	NMI	ARI	ACC	F1	NMI	ARI	ACC	F1
DeSE	+1	57.40	53.50	73.52	71.20	44.34	44.84	68.86	63.95	51.22	33.29	55.86	43.17	70.13	62.50	80.55	76.55
	+2	57.96	52.68	75.22	71.08	41.98	43.42	68.68	63.57	49.20	31.15	56.25	29.58	68.77	59.57	75.76	65.94
DMoN	+1	35.18	24.49	-	-	15.73	10.65	-	-	43.93	23.22	-	-	55.03	38.34	-	-
	+2	42.72	29.20	-	-	20.84	18.39	-	-	44.61	22.11	-	-	56.67	40.21	-	-
MinCut	+1	38.80	26.92	-	-	20.79	15.98	-	-	33.13	27.50	-	-	59.27	44.09	-	-
	+2	33.27	20.04	-	-	22.57	18.26	-	-	28.46	25.34	-	-	56.79	41.80	-	-
DGI	+1	53.62	44.70	-	-	38.87	38.62	-	-	25.98	15.10	-	-	30.60	17.71	-	-
	+2	56.05	47.72	-	-	38.12	37.53	-	-	32.42	18.22	-	-	23.55	11.36	-	-



(a) Cora 7 (NMI=56.26). (c) Cora 8 (NMI=57.40). (e) Cora 9 (NMI=57.96).

(b) Cora 7 true. (d) Cora 8 true. (f) Cora 9 true.

Figure 6: Robustness of cluster numbers on Cora.

is still able to identify them correctly. We attribute this to the use of SE loss and structural learning, which reduces the embedding distance between connected nodes and approaches cluster division from the perspective of overall structural stability. More analysis on the robustness of the cluster for the remaining three datasets is discussed in Appendix F.

We conduct experiments on three baselines for comparison. As shown in Table 6, we report the NMI, ARI, ACC, and F1 for DeSE, DMoN, MinCut, and DGI. ACC and F1 cannot be tested due to mismatched cluster numbers of baselines. It can be observed that when the number of clusters generated by the baselines slightly deviates from the ground truth, the performance of NMI and ARI declines sharply and exhibits instability. Moreover, due to the mismatch in the number of clusters with the ground truth, the baselines fail to provide ACC and F1 scores, which apply to all the baselines presented. This highlights their dependency on a predefined number of clusters. In contrast, our model DeSE does not suffer from this limitation. It not only maintains strong NMI and ARI performance even when the specified number of clusters deviates from the ground truth but also converges the number of clusters to the appropriate value, thereby achieving high ACC and F1 performance.

When the approximate range of the number of clusters is known in advance, as described above, DeSE achieves good clustering performance, and the expected number of clusters aligns well with the actual number. When the number of clusters is unknown, we can iteratively run DeSE to approximate convergence as shown in Table 7. Using the Cora dataset as a case study, we perform the following steps: We set the number of clusters to $c = N = 2708$,

Table 7: Case study on Cora when the number of clusters is unknown.

Round	Input: c	Output: NMI, clusters
1	$c = 2708$	NMI=39.40, clusters=372
2	$c = 372$	NMI=45.00, clusters=36
3	$c = 36$	NMI=51.17, clusters=14
4	$c = 14$	NMI=50.08, clusters=7
5	$c = 7$	NMI=57.96, clusters=7

obtaining an NMI of 39.40 with 372 clusters. We set $c = 372$ and repeat the experiment, yielding an NMI of 45.00 with 36 clusters. We set $c = 36$ and repeat the experiment, obtaining an NMI of 51.17 with 14 clusters. We then set $c = 14$ and repeat the experiment, resulting in an NMI of 50.08 with 7 clusters. We set $c = 7$ and repeat the experiment, achieving an NMI of 57.96 with 7 clusters. At this point, the preset number of clusters matches the output number, and we stop the testing process.

6 CONCLUSION

This paper presents a novel unsupervised graph clustering framework, DeSE, which incorporates deep structural entropy. The proposed framework addresses the challenges of structural quantification and structural learning to enhance clustering performance. We introduce a method for calculating structural entropy with soft assignment and design a Structural Learning Layer to optimize the original graph based on node features. Additionally, the Clustering Assignment Layer jointly learns node embeddings and a soft assignment matrix to derive node clusters through a new optimization approach that minimizes both SE loss and CE loss. Extensive experiments demonstrate the superiority and interoperability of DeSE while also showcasing its robustness in determining the number of clusters. Our findings highlight the potential of structural information theory in graph structure learning and may open new avenues for research on trainable soft assignment structural entropy in the integration of features and structure.

ACKNOWLEDGMENTS

This work has been supported by NSFC through grants 62322202, 62441612 and 62432006, Local Science and Technology Development Fund of Hebei Province Guided by the Central Government of China through grant 246Z0102G, the "Pioneer" and "Leading Goose" R&D Program of Zhejiang" through grant 2025C02044, National Key Laboratory under grant 241-HF-D07-01, Hebei Natural Science Foundation through grant F2024210008, and CCF-DiDi GAIA collaborative Research Funds for Young Scholars.

REFERENCES

[1] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. 2020. Spectral clustering with graph neural networks for graph pooling. In *International conference on machine learning*. PMLR, 874–883.

[2] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. 2020. Hierarchical representation learning in graph neural networks with node decimation pooling. *IEEE Transactions on Neural Networks and Learning Systems* 33, 5 (2020), 2195–2207.

[3] Yuwei Cao, Hao Peng, Angsheng Li, Chenyu You, Zhifeng Hao, and Philip S Yu. 2024. Multi-Relational Structural Entropy. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. 1–15.

[4] Yuwei Cao, Hao Peng, Zhengtao Yu, and Philip S Yu. 2024. Hierarchical and Incremental Structural Entropy Minimization for Unsupervised Social Event Detection. In *Proceedings of the AAAI conference on artificial intelligence*. 1–13.

[5] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. 2018. Harp: Hierarchical representation learning for networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[6] Kaize Ding, Yancheng Wang, Yingzhen Yang, and Huan Liu. 2023. Eliciting structural and semantic global knowledge in unsupervised graph contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 7378–7386.

[7] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. 2021. Slaps: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 22667–22681.

[8] Lei Gong, Sihang Zhou, Wenxuan Tu, and Xinwang Liu. 2022. Attributed Graph Clustering with Dual Redundancy Reduction.. In *IJCAI*. 3015–3021.

[9] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 66–74.

[10] Zhao Kang, Chong Peng, Qiang Cheng, Xinwang Liu, Xi Peng, Zenglin Xu, and Ling Tian. 2021. Structured graph learning for clustering and semi-supervised classification. *Pattern Recognition* 110 (2021), 107627.

[11] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[12] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[13] Angsheng Li and Yicheng Pan. 2016. Structural information and dynamical complexity of networks. *IEEE TIT* 62, 6 (2016), 3290–3339.

[14] Xunkai Li, Youpeng Hu, Yaoqi Sun, Ji Hu, Jiyong Zhang, and Meixia Qu. 2020. A deep graph structured clustering network. *IEEE Access* 8 (2020), 161727–161738.

[15] Ya-Wei Eileen Lin, Ronald R Coifman, Gal Mishne, and Ronen Talmon. 2023. Hyperbolic diffusion embedding and distance for hierarchical representation learning. In *International Conference on Machine Learning*. PMLR, 21003–21025.

[16] Jia Liu, Maoguo Gong, Qiguang Miao, Xiaogang Wang, and Hao Li. 2017. Structure learning for deep neural networks based on multiobjective optimization. *IEEE transactions on neural networks and learning systems* 29, 6 (2017), 2450–2463.

[17] Yiwei Liu, Jiamou Liu, Zijian Zhang, Liehuang Zhu, and Angsheng Li. 2019. REM: From structural entropy to community structure deception. *Proceedings of the Advances in Neural Information Processing Systems* 32 (2019), 1–11.

[18] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. 2022. Towards unsupervised deep graph structure learning. In *Proceedings of the ACM Web Conference 2022*. 1392–1403.

[19] Tianshu Lyu, Yuan Zhang, and Yan Zhang. 2017. Enhancing the network embedding quality with structural similarity. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 147–156.

[20] Qi Mao, Li Wang, Steve Goodison, and Yijun Sun. 2015. Dimensionality reduction via graph structure learning. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 765–774.

[21] Nairouz Mrabah, Mohamed Bouguessa, Mohamed Fawzi Touati, and Riadh Ksantini. 2022. Rethinking graph auto-encoder models for attributed graph clustering. *IEEE Transactions on Knowledge and Data Engineering* 35, 9 (2022), 9037–9053.

[22] Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the national academy of sciences* 103, 23 (2006), 8577–8582.

[23] Hao Peng, Jingyun Zhang, Xiang Huang, Zhifeng Hao, Angsheng Li, Zhengtao Yu, and Philip S Yu. 2024. Unsupervised Social Bot Detection via Structural Information Theory. *ACM Transactions on Information Systems* 42, 6 (2024), 42.

[24] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[25] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. 2023. Graph clustering with graph neural networks. *Journal of Machine Learning Research* 24, 127 (2023), 1–21.

[26] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. In *International Conference on Learning Representations*.

[27] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17 (2007), 395–416.

[28] Yifei Wang, Yupan Wang, Zeyu Zhang, Song Yang, Kaiqi Zhao, and Jiamou Liu. 2023. User: Unsupervised structural entropy-based robust graph neural network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 10235–10243.

[29] Chunyu Wei, Jian Liang, Di Liu, and Fei Wang. 2022. Contrastive graph structure learning via information bottleneck for recommendation. *Advances in Neural Information Processing Systems* 35 (2022), 20407–20420.

[30] Scott White and Padhraic Smyth. 2005. A spectral clustering approach to finding communities in graphs. In *Proceedings of the 2005 SIAM international conference on data mining*. SIAM, 274–285.

[31] Junran Wu, Xueyuan Chen, Bowen Shi, Shangzhe Li, and Ke Xu. 2023. SEGA: Structural Entropy Guided Anchor View for Graph Contrastive Learning. In *Proceedings of the ICML*. PMLR, 1–20.

[32] Junran Wu, Xueyuan Chen, Ke Xu, and Shangzhe Li. 2022. Structural entropy guided graph hierarchical pooling. In *Proceedings of the International Conference on Machine Learning*. PMLR, 24017–24030.

[33] Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. 2022. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems* 35 (2022), 27387–27401.

[34] Shuchen Wu, Noémi Élteto, Ishita Dasgupta, and Eric Schulz. 2022. Learning Structure from the Ground up—Hierarchical Representation Learning by Chunking. *Advances in Neural Information Processing Systems* 35 (2022), 36706–36721.

[35] Xihong Yang, Cheng Tan, Yue Liu, Ke Liang, Siwei Wang, Sihang Zhou, Jun Xia, Stan Z Li, Xinwang Liu, and En Zhu. 2023. Convert: Contrastive graph clustering with reliable augmentation. In *Proceedings of the 31st ACM International Conference on Multimedia*. 319–327.

[36] Yingguang Yang, Qi Wu, Buyun He, Hao Peng, Renyu Yang, Zhifeng Hao, and Yong Liao. 2024. SeBot: Structural Entropy Guided Multi-View Contrastive Learning for Social Bot Detection. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

[37] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*. PMLR, 40–48.

[38] Zhenyu Yang, Ge Zhang, Jia Wu, Jian Yang, Quan Z Sheng, Hao Peng, Angsheng Li, Shan Xue, and Jianlin Su. 2023. Minimum entropy principle guided graph neural networks. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 114–122.

[39] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems* 31 (2018).

[40] Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang. 2021. Graph-revised convolutional network. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part III*. Springer, 378–393.

[41] Guangjie Zeng, Hao Peng, Angsheng Li, Zhiwei Liu, Chunyang Liu, S Yu Philip, and Lifang He. 2023. Unsupervised Skin Lesion Segmentation via Structural Entropy Minimization on Multi-Scale Superpixel Graphs. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 768–777.

[42] Xianghua Zeng, Hao Peng, Angsheng Li, Chunyang Liu, Lifang He, and Philip S. Yu. 2023. Hierarchical State Abstraction based on Structural Information Principles. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, Edith Elkind (Ed.). International Joint Conferences on Artificial Intelligence Organization, 4549–4557.

[43] Chengde Zhang, Yu Lei, Xia Xiao, and Xinzhong Chen. 2022. Cross-media video event mining based on attention graph structure learning. *Neurocomputing* 502 (2022), 148–158.

[44] Hongyuan Zhang, Pei Li, Rui Zhang, and Xuelong Li. 2022. Embedding graph auto-encoder for graph clustering. *IEEE Transactions on Neural Networks and Learning Systems* 34, 11 (2022), 9352–9362.

[45] Tianqi Zhang, Yun Xiong, Jiawei Zhang, Yao Zhang, Yizhu Jiao, and Yangyong Zhu. 2020. CommDGI: community detection oriented deep graph infomax. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 1843–1852.

[46] Huachi Zhou, Shuang Zhou, Keyu Duan, Xiao Huang, Qiaoyu Tan, and Zailiang Yu. 2023. Interest driven graph structure learning for session-based recommendation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 284–296.

[47] Dongcheng Zou, Hao Peng, Xiang Huang, Renyu Yang, Jianxin Li, Jia Wu, Chunyang Liu, and Philip S Yu. 2023. SE-GSL: A General and Effective Graph Structure Learning Framework through Structural Entropy Optimization. In *Proceedings of the ACM Web Conference 2023*. 499–510.

[48] Dongcheng Zou, Senzhang Wang, Xuefeng Li, Hao Peng, Yuandong Wang, Chunyang Liu, Kehua Sheng, and Bo Zhang. 2024. Multispans: A multi-range spatial-temporal transformer network for traffic forecast via structural entropy optimization. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 1–10.

A NOTATIONS

The comprehensive list of the primary symbols used throughout this paper is presented in Table 8.

Table 8: Forms and interpretations of notations.

Symbol	Definition
$G = (V, \mathcal{E}, X)$	The original graph.
$A_g \in \{0,1\}^{N\times N}$	The adjacency matrix from the original graph.
$A_f \in \{0,1\}^{N\times N}$	The adjacency matrix from the learned graph.
$V, \mathcal{E}$	The node/edge set.
$X \in R^{N\times f}$	The node feature matrix.
$S \in \{0,1\}^{N\times c}$	The node assignment matrix.
$S^k \in R^{N_k\times N_{k-1}}$	The assignment matrix between layer k and layer $k-1$.
$C^k \in R^{N\times N_k}$	The direct assignment matrix between nodes and layer k.
N, N_k	The number of nodes, The number of vertices of layer k.
M, c	The number of edges/clusters.
f, d	The dimension of feature/embedding.
$\mathcal{T}; \lambda$	Encoding tree; The root vertice of the encoding tree.
$\alpha; \alpha^-$	Vertice on encoding tree. Parent vertice of vertice α.
T_α	A node subset corresponds to vertice α.
g_α	Number of cutting edges of nodes in vertice α.
h	Height of the encoding tree.
$vol(G); vol(\alpha)$	Volume of Graph G; Volume of vertice α.
$vol^k[i]$	Volume of vertex i at layer k with soft assignment.
$vol^k_{in}[i]$	Internal volume of vertex i at layer k.
g^k_i	Cutting edges of vertex i at layer k with soft assignment.
$H^{\mathcal{T}}(G)$	The structural entropy.
$H^{\mathcal{T}}_{sa}(G)$	The soft assignment SE of graph G.
$H_k(G)$	The k-dimensional structure entropy.
$H_{sa}(G;k)$	The soft assignment structural entropy at layer k.
$D \in R^N$	The degree vector for all nodes.
$W \in R^{N\times N}$	The weight matrix.
$MLP(;\Theta_f)$	The multilayer perceptron with parameter Θ_f.
$KNN(;K)$	The k-nearest nerighbors algorithm with parameter K.
$GNN_{emb}(;\Theta_1)$	The embedding learner with parameter Θ_1.
$GNN_{ass}(;\Theta_2)$	The soft assignment learner with parameter Θ_2.
Γ	The attention matrix.
$\mathcal{L}_{ce}, \mathcal{L}_{se}$	The CE loss. The SE loss.
$\lambda_{ce}, \lambda_{se}$	The coefficients of CE loss and SE loss.
β_f, K	The weight of A_f. The number of neighbors in KNN.

B ALGORITHM

The algorithm of DeSE is summarized in Algorithm 1.

C BASELINES

Detailed descriptions of 8 baselines compared to our work are introduced as follows:

• **DMoN** [25] introduces a modularity measure of clustering quality to optimize cluster assignment in an end-to-end manner and proposes Deep Modularity Networks.

• **MinCut** [1] formulates a continuous relaxation of the normalized minCUT problem and trains a GNN to compute cluster assignments by optimizing this objective.

• **DGI** [26] is a versatile method for learning node representations in graph-structured data based on maximizing the mutual information between local patch representations and high-level graph summaries.

• **SUBLIME** [18] is a structure bootstrapping contrastive Learning framework with the aid of self-supervised contrastive learning, where the learned graph topology is optimized by data itself.

• **EGAE** [44] is a specific GAE-based model for graph clustering that is consistent with the theory of learning well-explainable representations.

• **CONVERT** [35] is a contrastive graph clustering network with reliable augmentation and distills reliable semantic information by recovering the perturbed latent embeddings.

• **AGC-DRR** [8] is an attributed graph clustering framework with dual redundancy reduction to reduce the information redundancy in both input and latent feature space.

• **RDGAE** [21] is a tailored GAE model that triggers a correction mechanism against Feature Drift by gradually transforming the reconstructed graph into a clustering-oriented one.

Algorithm 1: Algorithm of one epoch of DeSE.

Input: Original adjacency matrix: A_g; Feature matrix: X; Coefficient of CE and SE loss: $\lambda_{ce}, \lambda_{se}$; Weight of A_f: β_f; Number of neighbors: K; Number of clusters: c; Dimension of embedding: d; Number of layers: h.

Output: Node assignment matrix: $S \in \{0,1\}^{N\times c}$.

// SLL

Initialize A_f with structure learning layer via Eq. 7;

Update A_f via Eq. 8;

Calculate W_0 with β_f, A_g, and A_f via Eq. 9;

Initialize embedding E with GNN and W;

Initialize list $S_{list} \leftarrow \{\}$;

// ASS

for *k=1,2,..,h* **do**

  Calculate embedding H with Embedding Learner GNN_{emb} via Eq. 10;

  Calculate matrix S^k with Soft AssignmentLearner GNN_{ass} via Eq. 11;

  Update A_g and A_f via Eq. 12;

  Calculate W_k with β_f, A_g, and A_f via Eq. 9;

  Store S^k to S_{list};

end

// Soft Assignment SE

Initialize $\mathcal{L}_{se} \leftarrow 0$;

for *k=1,2,..,h* **do**

  Calculate C^k with $\{S^h, ..., S^{k+1}\}$ via Eq. 3;

  Calculate vol^k via Eq. 4;

  Calculate g^k via Eq. 5;

  Calculate $H_{sa}(G;k)$ with S^k, C^k, vol^k, and g^k via Eq. 6;

  Update $\mathcal{L}_{se} \leftarrow \mathcal{L}_{se} + H_{sa}(G;k)$;

end

Calculate $\mathcal{L}_{ce}$ with embedding E via Eq. 13;

Calculate final loss $\mathcal{L}$ with coefficients $\lambda_{ce}, \lambda_{se}$ via Eq. 14;

// hard assignment matrix

Initialize $S \leftarrow 0^{N\times c}$;

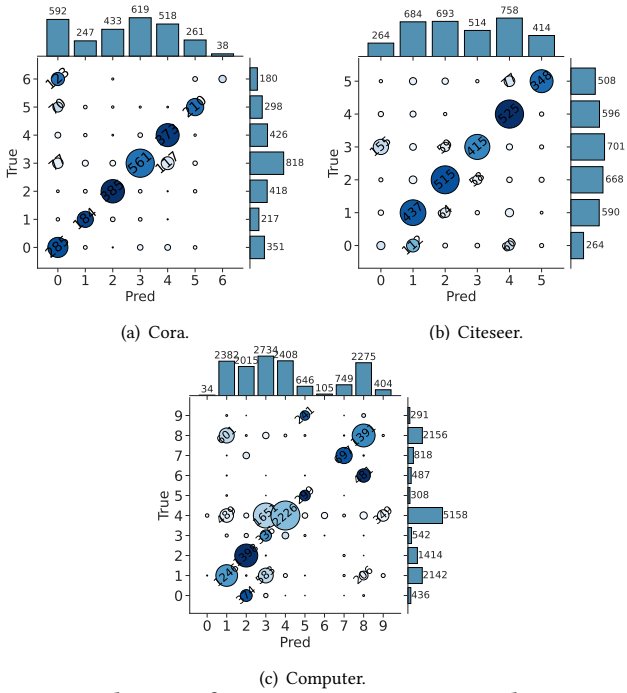
Calculate the index idx of the maximum value of C^k along each row;

for *i=1,2,...,N* **do**

  Set $S[i, idx[i]] = 1$;

end

Return the hard assignment matrix S.



(a) Cora. (b) Citeseer.

(c) Computer.

Figure 7: Clusters of DeSE on Cora, Citeseer, and Computer datasets.

D DATASET

Table 2 provides computed values like the average number of edges per node (Sparsity) and the number of isolated nodes (Iso.). Detailed descriptions of four datasets are provided below: • **Cora** [37] is a citation network composed of research papers in the field of machine learning, with each paper linked to others that it cites. The nodes represent the papers, while the edges denote citation relationships between them. • **Citeseer** [37] is a citation network similar to Cora, consisting of scientific papers. Each node represents a research paper, and the edges represent citation links between them. • **Computer** [24] is part of the Amazon co-purchase graph, where nodes represent products in the "computers" category on Amazon, and edges indicate that two products are frequently bought together. • **Photo** [24] is a part of the Amazon co-purchase graph, but it focuses on products in the "photo" category (e.g., cameras, photography accessories). Similar to the Computer dataset, nodes represent products, and edges indicate frequent co-purchases.

E GRAPH CLUSTERING PERFORMANCE

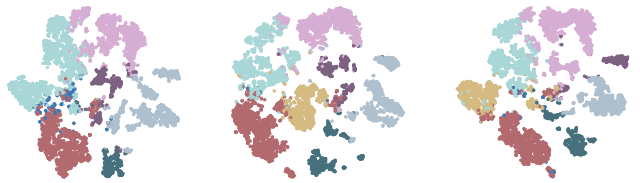
The detailed correspondence between the nodes of the predicted clusters and the true clusters of DeSE on Cora, Citeseer, and Computer datasets are presented in Figure 7. We can observe that DeSE demonstrates concentrated and accurate predictions for larger clusters across the three datasets, while smaller clusters are often disregarded. For example, in Cora, the predicted cluster 0 primarily contains most of the actual clusters 0 and 6, but since cluster 0 is larger in size, the predicted cluster is classified as cluster 0. Similarly, in Citeseer, the predicted cluster 0 mostly contains actual cluster 3, but because the predicted cluster 3 includes more of the actual cluster 3, it is not assigned to cluster 3. In contrast, there are more misclassifications in the Computer dataset, particularly with the actual cluster 4 being more widely dispersed in the predictions. We believe that the primary reason for these errors is the unclear boundaries between clusters. As seen in Figure 7, the errors tend to appear collectively, indicating that smaller clusters are easily merged into a larger cluster or that a large cluster is split into smaller ones. Improving or fine-tuning cluster boundaries within DeSE is the next step for future research.

F ROBUSTNESS ON CLUTSERS

(a) cluster6 (NMI=42.16). (c) cluster7 (NMI=44.34). (e) cluster8 (NMI=41.98).

(b) cluster6 true. (d) cluster7 true. (f) cluster8 true.

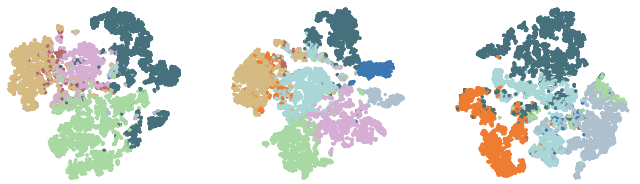
Figure 8: Robustness of cluster numbers on Citeseer.



(a) cluster8 (NMI=63.04). (c) cluster9 (NMI=70.13). (e) cluster10 (NMI=68.78).

(b) cluster8 true. (d) cluster9 true. (f) cluster10 true.

Figure 9: Robustness of cluster numbers On Photo.



(a) cluster10 (NMI=47.63). (c) cluster11 (NMI=51.22). (e) cluster12 (NMI=49.20).

(b) cluster10 true. (d) cluster11 true. (f) cluster12 true.

Figure 10: Robustness of cluster numbers On Computer.

The visual results of our model DeSE on Citeseer, Photo, and Computer datasets about the robustness of clusters are presented in Figure 8, Figure 9, and Figure 10. We also observe that, regardless of the number of clusters set (as long as it is greater than or equal to the original number), the model consistently forms the same number of clusters adaptively. While there are some differences in NMI, this is due to the lack of fine-tuning of parameters for different cluster numbers. A key direction for future work is to improve the model to reduce the impact of hyperparameters on clustering accuracy across varying numbers of clusters.

G SENSITIVITY ANALYSIS

Table 9 presents the ACC results of DeSE with λ_{se} in the range of $\{0.01, 0.05, 0.2, 0.5\}$ and λ_{ce} in the range of $\{0.1, 0.5, 1, 5\}$ as a supplement to Section 5.4 "Coefficients of SE loss and CE loss λ_{se} and λ_{ce}". It can be observed that, for ACC, the optimal parameter selection follows a similar trend to NMI in Table 5. Specifically, the SE loss tends to favor smaller coefficients, while the CE loss prefers larger coefficients. Despite its smaller value, the SE loss plays a significant role in improving clustering accuracy.

H TIME AND MEMORY ANALYSIS

We set the epoch to 600 and conduct 10 experiments on DeSE. Table 10 records the average runtime across the four datasets. It can be observed that the DeSE's runtime increases with the number of nodes and edges in the dataset, especially for the Computer dataset, which has higher "Sparsity". However, overall, the runtime remains within an acceptable range. Future efficiency improvements may be achievable through enhancements in the selection of K-nearest neighbors in large-scale graphs and the computation of soft-assignment structural entropy. In addition, Table 11 shows that the hyperparameter memory usage of DeSE is also a major advantage.

Table 9: Sensitivity of hyperparameters λ_{se} and λ_{ce} with ACC.

Variation	**SE loss λ_{se}**				**CE loss λ_{ce}**			
	0.01	0.05	0.2	0.5	0.1	0.5	1	5
Cora	**75.22**	74.45	72.90	72.08	72.30	74.59	74.63	**75.22**
Citeseer	**68.86**	52.30	51.52	50.86	67.96	68.86	68.89	**69.04**
Computer	43.20	-	**55.87**	50.49	54.46	**55.87**	44.18	43.70
Photo	**80.55**	80.08	71.16	66.77	66.76	71.71	72.18	**80.55**

Table 10: Average time cost for DeSE and baselines (Sec).

Method	Cora	Citeseer	Computer	Photo
DeSE	65.87	81.17	2037.16	684.54
DMoN	65.07	77.80	293.47	179.90
MinCut	69.56	79.18	243.15	146.48
DGI	178.24	250.88	1144.88	581.18
SUBLIME	104.80	137.21	477.92	215.87
EGAE	106.17	67.52	506.03	184.83
CONVERT	91.85	146.44	281.85	140.51
AGC-DRR	234.44	533.16	8096.46	2885.40
RDGAE	56.03	107.47	1284.25	580.86

Table 11: Memory usage analysis for DeSE and baselines (MB).

Method	Cora	Citeseer	Computer	Photo
DeSE	0.09	0.92	0.21	0.43
DMoN	0.96	2.07	0.63	0.62
MinCut	0.96	2.07	0.63	0.62
DGI	5.81	10.24	4.51	4.47
SUBLIME	2.96	7.39	1.66	1.61
EGAE	1.52	3.74	0.87	0.85
CONVERT	14.69	51.58	5.75	5.75
AGC-DRR	6.11	6.11	6.11	6.11
RDGAE	0.18	0.45	0.10	0.09