Annals of Emerging Technologies in Computing (AETiC)
Vol. 10, No. 3, 2026

Research Article

Graph Clustering Research Based on Adaptive and Optimal Probabilistic Graph Learning

Lingguo Zou[1] and Meihua Zhang[2,*]

[1]Xiamen Ocean Vocationnal College, Xiamen China
zoulingguo@xmoc.edu.cn; meihuazhang17@outlook.com
[2]Xiamen Huatian International Vocation Institute, Xiamen China
*Correspondence: meihuazhang17@outlook.com

Received: 26 December 2025; Accepted: 30 June 2026; Published: 1 July 2026

Abstract: With the continuous development of mobile internet, related data has experienced explosive growth. However, existing graph clustering methods suffer from issues such as high sensitivity to the quality of initial graph structures, the disconnect between graph construction and representation learning, and difficulty in handling graph structural noise. This research proposes a graph clustering model based on adaptive graph learning and optimal probabilistic graph learning. First, to address the limitation of static graph structures that cannot dynamically optimize based on node representations, the Adaptive Graph Clustering (AGC) model is proposed. Through a learnable adjacency matrix update mechanism, the graph structure adapts to serve the clustering objective during training. Second, to address the limitation of existing deterministic graph learning ignoring edge uncertainty, the Optimal Probabilistic Graph Clustering (OPGC) model is proposed. It models the graph structure as a Bernoulli distribution, learning edge existence probabilities through probability-weighted graph convolutions and entropy regularization, thereby explicitly quantifying structural noise. Additionally, a dual self-supervised clustering strategy is designed to simultaneously generate soft assignments from both attribute and graph spaces while minimizing Kullback-Leibler divergence from the target distribution. This unifies attribute feature learning and probabilistic graph structure learning within an end-to-end framework. The adaptive graph clustering model demonstrates outstanding performance on both structural and textual data. It achieves a maximum clustering accuracy of 80.2%, surpassing the second-best deep adaptive graph clustering by 7.8%. After removing 15% of original edges, the model's normalized mutual information decreases by only 0.044. The optimal probabilistic graph clustering model achieves a clustering accuracy of 0.852. Ablation experiments demonstrate that the adaptive graph learning module is most critical for enhancing model performance. The model converges 26.9% faster than graph attention networks and consumes 23.6% less memory. Consequently, the proposed model effectively improves graph clustering accuracy and robustness, significantly boosts computational efficiency, and simplifies complex networks.

Keywords: ***Adaptive graph learning; Graph clustering; Graph convolutional neural network; Optimal probability; Self-supervised learning***

1. Introduction

Graph clustering, a key task in complex network analysis, aims to partition networks into groups characterized by strong intra-class connectivity and sparse inter-class connections by identifying closely

Lingguo Zou and Meihua Zhang, "Graph Clustering Research Based on Adaptive and Optimal Probabilistic Graph Learning", *Annals of Emerging Technologies in Computing (AETiC)*, Print ISSN: 2516-0281, Online ISSN: 2516-029X, 1 July 2026, Vol. 10, No. 3, pp. 67-83, Published by International Association for Educators and Researchers (IAER), DOI: 10.33166/AETiC.2026.03.005, Available: http://aetic.theiaer.org/archive/v10/v10n3/p5.html.

connected nodes [1]. In recent years, graph clustering has been extensively applied in fields such as community detection in social networks, topic mining in citation networks, functional module identification in biological networks, and pathway analysis in gene regulatory networks [2-3]. These applications not only help reveal latent community structures and functional modules within networks but also enable deeper exploration of complex inter-node relationships, providing crucial support for understanding the organizational principles and behavioral patterns of complex systems [4-5]. However, existing methods still face three critical challenges: (1) High sensitivity to the quality of the initial graph structure. Real-world graph data often contains noise, missing edges, and spurious connections. Existing methods (such as graph autoencoders and graph convolutional networks) learn directly on the raw graph, leading to performance that heavily depends on graph quality. (2) Rigid and inflexible graph structures. Most approaches employ static adjacency matrices, unable to dynamically optimize graph structures based on node representations, thereby limiting the model's ability to capture complex relationships. (3) Disconnect between graph construction and representation learning. The traditional two-step framework (first graph construction, then clustering) prevents graph construction from serving clustering objectives, making it difficult to obtain clustering-friendly representations. To address these challenges, this study proposes a graph clustering framework based on adaptive and optimal probabilistic graph learning. Key contributions include: (1) Proposing the Adaptive Graph Clustering (AGC) model, which employs a dynamic graph structure learning mechanism. This enables the graph adjacency matrix to adaptively optimize and update based on node representations, overcoming the limitations of static graph structures. (2) Introducing the Optimal Probabilistic Graph Clustering (OPGC) model. This model explicitly models the uncertainty of edge existence through a probability distribution-based neighbor assignment algorithm and graph reconstruction regularization, enhancing the model's robustness to noise. (3) Designs a dual self-supervised clustering strategy that unifies attribute feature learning and graph structure learning within an end-to-end training framework, improving clustering performance through collaborative optimization guided by pseudo labels. (4) Conducts systematic experiments on multiple benchmark datasets, validating the superiority of the proposed methods in clustering accuracy, robustness, and computational efficiency.

2. Literature Review

Early work on graph clustering primarily relied on spectral methods. Graph Convolutional Networks (GCNs) extended convolutions to graph data, learning node embeddings through neighborhood aggregation and providing an end-to-end framework for graph clustering [6]. However, GCNs implicitly assume that the input graph structure is complete and accurate, faithfully reflecting node relationships. In real-world scenarios, this faces severe challenges such as fake followers in social networks, missing citations in citation networks, and measurement errors in biological networks—all of which can cause the initial adjacency matrix to deviate from true relationships. This limitation has driven two research directions: attention mechanisms to distinguish neighbor importance, and dynamic structure learning to correct the initial graph. To address GCN's equal weighting of all neighbors, Graph Attention Networks (GAT) introduced attention mechanisms to assign differentiated weights to distinct neighbors. Wang *et al.*'s Deep Neighbor-Aware Embedding method employs an attention encoder to absorb information from neighbors and an inner-product decoder to reconstruct the graph structure. This approach mitigates the negative impact of heterogeneous edges to some extent [7]. However, attention mechanisms only adjust weights on a fixed graph structure and cannot fundamentally correct erroneous connections. When the initial graph contains substantial noise, even optimal attention allocation struggles to recover performance losses. This insight drove a paradigm shift from "weight adjustment" to "structure learning." Tsitsulin *et al.* proposed a deep modular network inspired by the modularity metric of clustering quality, constructing an unsupervised pooling architecture that optimizes graph structures through end-to-end learning [8].

The core idea of dynamic structure learning is that the graph adjacency matrix should not be a fixed input but a learnable variable. Yang *et al.* proposed a fuzzy-based deep attribute graph clustering model, employing a graph convolutional network to encode both the graph structure and node attributes. They introduced a self-supervised training strategy to guide nodes toward cluster centers [9]. While these approaches achieve preliminary joint optimization of graph structure and node representations, two key limitations remain: (1) Graph learning is modeled as deterministic optimization, ignoring edge existence uncertainty; (2) Structural

updates may accelerate over-smoothing, causing node representations to converge. Mrabah *et al.* revealed an inherent contradiction in dynamic structure learning: while dynamically adding edges corrects initial noise, dense connections accelerate over-smoothing—as GCN layers increase, node representations become increasingly uniform and lose discriminative power [10]. Liu *et al.*'s Dual Correlation Reduction Network employs a shallow architecture to avoid over-smoothing by forcing cross-view samples and feature correlation matrices to reduce information correlation, yet this limits the model's ability to capture higher-order structural information [11]. The tension between over-smoothing and structural learning has become a research focus: how to preserve node distinctiveness while correcting graph structures? Some studies explore sparse constraints to limit edge additions, but sparsity settings rely on empirical judgment; another approach introduces probabilistic modeling to maintain structural diversity by learning edge probability distributions. To mitigate reliance on labeled data, self-supervised learning has emerged as a key paradigm. Liu *et al*. proposed a simple contrastive graph clustering algorithm that employs low-pass denoising to preprocess neighbor information and designs a cross-view structural consistency objective function, achieving speeds 7 times faster than contrastive deep clustering methods across 7 benchmark datasets. Ju *et al.* extended contrastive learning to graph-level clustering, introducing a graph-level contrastive clustering framework that integrates instance-level and cluster-level contrastive learning [12].

However, contrastive learning methods face fundamental challenges: (1) Negative sample construction is complex and sensitive to quantity; (2) Graph design lacks theoretical guidance and may disrupt core structures. Yang *et al*.'s spatial feature map contrastive learning method generates negative sample embeddings via row shuffling, eliminating the need for negative sample generators and simplifying the contrastive learning process. Yet it remains insufficient in graph structure learning [13]. Sieranoja *et al*. enhanced clustering performance through algorithmic optimization. Their M-algorithm optimizes the cost function by randomly merging partitioned clusters, offering insights for designing graph clustering objective functions [14].

Based on this critical analysis, key gaps in existing research can be precisely identified: 1: Deterministic graph learning ignores uncertainty. Existing dynamic structure learning methods model graph learning as deterministic optimization, obtaining sparse graphs through L_1 norm or low-rank constraints. While effective in low-noise scenarios, deterministic decisions may misinterpret noise as genuine structure when noise levels increase. Probabilistic graph modeling can identify uncertain connections via variance estimation, but existing probabilistic approaches are often based on fixed graph structures and fail to integrate with dynamic learning. 2: Disconnect between structure learning and clustering objectives. Most dynamic graph learning methods prioritize graph reconstruction, aiming to approximate the original graph. This assumption compromises performance when the original graph contains substantial noise. Ideal graph learning should serve clustering objectives—the learned graph should densely connect similar nodes and sparsely connect dissimilar ones, rather than merely fitting the initial input. 3: Unresolved tension between over-smoothing and structural updates. Dynamically adding edges may accelerate over-smoothing. Existing methods mitigate this through layer limits or sparsity constraints, but lack theoretical guarantees. A probabilistic perspective is needed: Can dynamic adjustment of edge weights dynamically regulate information flow, balancing structural refinement with discriminative preservation? 4: Computational complexity and view dependency of contrastive learning. Contrastive learning relies on complex view design and negative sample sampling, incurring high computational overhead on large-scale graphs.

Compared to existing adaptive graph clustering methods (such as DAEGC), the AGC model proposed in this paper directly couples graph structure updates with the clustering objective. Through a learnable adjacency matrix and a comprehensive loss function (incorporating sparsity, low-rank, and Laplacian regularization), the graph structure dynamically serves the clustering process during training, rather than merely fitting the original graph. Compared with existing probabilistic graph clustering methods, the OPGC model models the graph structure as a Bernoulli-distributed random variable, simultaneously optimizing edge presence probabilities and node representations within an adaptive learning framework, and explicitly quantifies structural noise, rather than employing deterministic graph updates or predefined fixed-probability graphs. Compared to existing self-supervised contrastive clustering methods (such as DGI and GRACE), this paper designs a negative-sample-free and multi-view-enhancement-free dual KL-divergence self-supervised strategy that simultaneously generates soft assignments in both the attribute space and the graph structure space and aligns

them with the target distribution with high confidence, thereby avoiding the computational overhead of negative sample sampling and the hyperparameter sensitivity issues associated with view design. These innovations achieve end-to-end unification of adaptive graph learning, probabilistic graph modeling, and self-supervised clustering, significantly improving noise robustness and clustering accuracy while maintaining linear computational complexity.

3. Materials and Methods

3.1. Graph clustering based on adaptive graph learning

Existing advanced graph clustering methods include GAEs, contrastive learning, and graph convolutional networks, but these methods all require high quality initial graphs [15]. The original GS contains a lot of noise and lacks flexibility. Existing methods cannot precisely describe the relationships between graph nodes. Therefore, the AGC model was proposed. The model consists of three modules, and the specific structure is denoted in Figure 1.

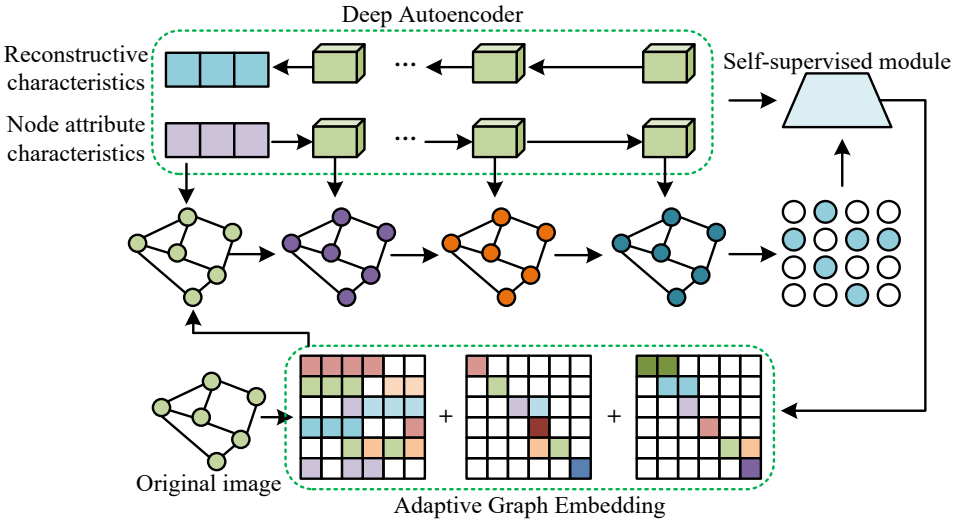

Figure 1. Specific structure of the adaptive graph clustering model

In Figure 1, the AGC model comprises three core modules: a deep autoencoder, adaptive graph embedding, and self-supervision. Through end-to-end joint training, it achieves synergistic optimization of attribute feature learning and graph structure learning. First, the deep autoencoder module encodes and compresses raw high-dimensional features. It nonlinearly projects input data into a low-dimensional discriminative space through multi-layer graph convolutional networks, then reconstructs original features via a symmetric decoder. By minimizing reconstruction loss, it extracts intrinsic, highly discriminative essential features from the data. Simultaneously, the adaptive graph embedding module abandons fixed initial graph structures, adopting a dynamic graph learning mechanism. By optimizing a composite loss function incorporating structural fidelity, sparsity constraints, graph Laplacian regularization, and feature smoothing regularization, it enables adaptive updates to the graph adjacency matrix based on node representations. This updated graph structure is then utilized to generate high-quality structured embedding representations through the graph convolutional network. Finally, the self-supervised module concurrently processes attribute representations from the autoencoder and structural representations from the graph convolutional network. It calculates soft assignment probabilities for each sample to its respective cluster center. By squaring and normalizing the attribute assignment probabilities, it generates a higher-confidence target distribution. Minimizing the Kullback-Leibler divergence between attribute assignments and the target distribution, as well as between structural assignments and the target distribution, achieves dual self-supervised alignment. This unified training framework guides mutually reinforcing attribute learning and graph structure learning,

ultimately generating clustering-friendly data representations that preserve intrinsic attributes while integrating topological structure. The encoder learns the latent representation through l layer nonlinear transformation, where the l layer representation is shown in Equation (1) [16].

$$H_l^e = \vartheta\left(\overline{A}H_{l-1}^e W_l + b_l^e\right) \tag{1}$$

In Equation (1), H_l^e represents the representation of the encoder at layer l, ϑ means the nonlinear activation function, $\overline{A}$ indicates an adaptive adjacency matrix, W_l represents the weight coefficient for the $l-1$ layer of the encoder. H_{l-1}^e represents the representation of the encoder at layer $l-1$, and b_l^e means the bias term of the encoder at layer l. The decoding process requires symmetrical reconstruction of the initial data, and the decoding representation of layer l is shown in Equation (2).

$$H_l^d = \vartheta\left(\overline{A}^T H_{l-1}^d W_l' + b_l^d\right) \tag{2}$$

In Equation (2), H_l^d refers to the representation of the l th layer of the decoder, $\overline{A}^T$ denotes the transpose of the adaptive adjacency matrix, W_l' represents the weight coefficient for the $l-1$ layer of the decoder. H_{l-1}^d represents the representation of the $l-1$ th layer of the decoder, and b_l^d represents the bias term of the l th layer of the decoder. The specific structure of the depth autoencoder is shown in Figure 2.

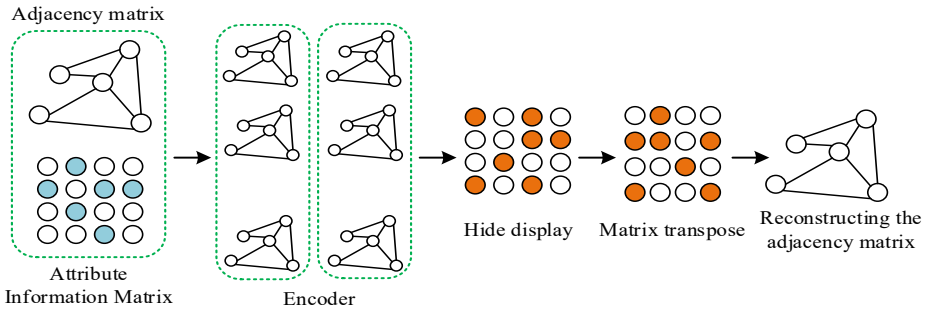

Figure 2. Specific architecture of a deep autoencoder

In Figure 2, the encoder includes two graph convolutional networks, each with l layers. The first graph convolutional network receives the input data and the adjacency matrix, generating an intermediate representation. The second graph convolutional network further processes the output of the first layer, generating hidden representations of the nodes. The hidden layers capture the attributes and structure of the input graph. Minimizing the reconstruction loss between the initial input data and the reconstructed output enhances the features' discriminative and reconstructive capabilities. This loss function is defined as shown in Equation (3).

$$L_{Rloss} = \frac{1}{2N}\left\|H_0^e + H_l^d\right\|_F^2 \tag{3}$$

In Equation (3), L_{Rloss} represents the reconstruction loss, N denotes the total amount of samples in the dataset, H_0^e means the initial input data, and F represents the Frobenius norm. The study employs an adaptive graph embedding module to dynamically learn the graph adjacency matrix, replacing the original GS, thereby improving the discriminative power of the embedded representation. The specific structure of the dynamic graph learning is denoted in Figure 3.

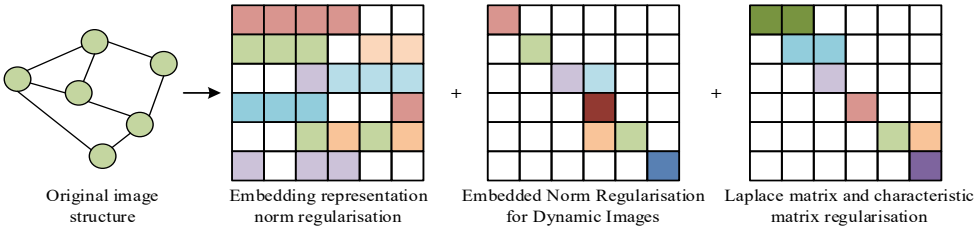

Figure 3. Specific structure of dynamic image learning

In Figure 3, the dynamic graph learning structure is responsible for processing and learning the dynamic changes of graph data. The input is the original GS, which contains information about nodes and edges. It is converted into an embedding representation through a graph convolutional network and then a dynamic graph embedding is generated. It is then regularized, including regularizing the embedding representation and the dynamic graph embedding norm, as well as the Laplacian matrix and the feature matrix. Finally, the dynamic graph embedding is updated to generate a new graph embedding [17]. The goal of learning the graph is to make it structurally close to the original graph, while possessing ideal low-rank and sparse properties, and reflecting the local connectivity of node features. The comprehensive loss function for this goal is calculated as denoted in Equation (4).

$$\min L_D = \|D+O\|_F^2 + \delta\|D\|_1 + \lambda\|D\|_* + \kappa Tr\left(X^T \tilde{l}_D X\right) \tag{4}$$

In Equation (4), L_D represents the comprehensive loss function, D represents the learning graph, O represents the original graph, δ and λ represent the hyperparameters of the sparsity and low-rank constraints, respectively, κ represents the balance parameter, Tr refers to the trace of the matrix, i.e., the sum of all elements on the main diagonal of the matrix. X indicates the node feature matrix, T means the transpose of the matrix, and $\tilde{l}_D$ represents the normalized graph Laplacian matrix. The learned dynamic graph is used to generate the embedded representation. Directly optimizing the full adjacency matrix would introduce $O(N^2)$ computational complexity, which is infeasible when handling large-scale graph data (e.g., containing hundreds of thousands of nodes). To address this issue, the AGC model does not compute similarity for all node pairs in each iteration. Instead, it dynamically maintains a k-sized neighborhood set for each node based on the current node. The model optimizes only the connection weights with these k nearest neighbors, while the weights of all other edges are forced to zero. This k-NN-based sparsification strategy reduces the complexity of each graph update from $O(N^2)$ to O(N log N), enabling the model to scale to larger graph datasets. In the OPGC model, probabilistic graph construction similarly relies on this k-NN approximation, modeling only the candidate neighbor set with a Bernoulli distribution. This avoids performing probability calculations across the entire graph. The input of each graph convolution layer integrates the structural representation of the previous layer and the attribute feature representation of the autoencoder, as shown in Equation (5).

$$\tilde{C}_{l-1} = (1-w)C_{l-1} + wH_{l-1} \tag{5}$$

In Equation (5), $\tilde{C}_{l-1}$ means the input of the l th layer of the graph convolutional network, w represents the weight coefficient, C_{l-1} represents the structural representation of the graph, and H_{l-1} means the latent representation of the basic information of the graph. This study uses the fused representation $\tilde{C}_{l-1}$ as input and generates new embeddings through dynamic graph convolutional layers. The self-supervised clustering (SSC) module unifies the deep autoencoder module and the adaptive learning module into an end-to-end framework through a self-generated pseudo-label mechanism, guiding them to learn cluster-friendly representations. The similarity calculation between the deep autoencoder output and the cluster centers is shown in Equation (6).

$$Q_{H_{ij}} = \frac{1}{\sum_k \left(1+\left\|h_i-\alpha_k\right\|_2^2\right)^{-1} \cdot \left(1+\left\|h_i-\alpha_j\right\|_2^2\right)} \tag{6}$$

In Equation (6), $Q_{H_{ij}}$ represents the similarity value, h_i stands for the latent vector of the sample i, α_j denotes the center vector of the cluster j, and α_k means the center vector of the cluster k. A sharper target distribution is obtained by normalizing $Q_{H_{ij}}$ by squares, and it is used as a supervision signal, as shown in Equation (7).

$$P_{H_{ij}} = \frac{Q_{H_{ij}}^2}{\sum_i Q_{H_{ij}}} \div \sum_J \left(\frac{Q_{H_{ij}}^2}{\sum_i Q_{H_{ij}}} \right) \tag{7}$$

In Equation (7), $P_{H_{ij}}$ represents the optimized probability distribution. The study uses the minimization of the KL divergence between the target distribution $P_{H_{ij}}$ and the Soft Assignment (SA) of the two modules to achieve self-supervised alignment, as shown in Equation (8).

$$L_{scl} = KL\left(P_H \| Q_H\right) + KL\left(P_H \| Q_Z\right) \tag{8}$$

In Equation (8), L_{scl} represents the SSC loss function, KL means the divergence, and the divergence KL is 0 when the two distributions are exactly the same. Q_Z represents the output of the adaptive graph embedding model.

3.2. Graph clustering based on optimal probabilistic graph learning

Although the AGC model effectively alleviates its dependence on the initial GS through adaptive graph learning, there is still room for improvement in handling heterogeneous data and optimizing the graph construction process. Existing graph clustering methods are not only affected by the quality of the initial image, but also suffer from poor clustering performance due to the separate graph construction and representation learning processes. Therefore, this study employs OPGL to construct a weighted adjacency graph for all samples, further improving the graph representation learning performance. The specific structure of the proposed OPGC model is denoted in Figure 4.

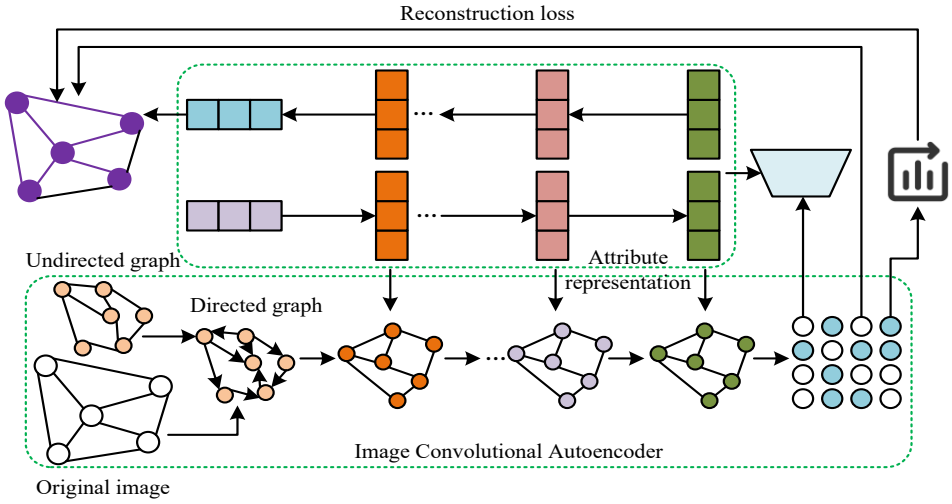

Figure 4. Specific structure of the OPGC model

In Figure 4, the OPGC model introduces a probabilistic graph learning mechanism based on AGC. It consists of three modules: an attribute autoencoder, a graph convolutional autoencoder, and a dual self-supervision module. By explicitly modeling the uncertainty of edge existence, the model enhances its robustness against noise. First, the attribute autoencoder module encodes and reconstructs raw high-dimensional features to extract pure low-dimensional attribute representations. These representations serve not only for subsequent clustering but also as input features for probabilistic graph learning. Subsequently, the Graph Convolutional Autoencoder module employs a probabilistic graph learning strategy. It treats the graph structure as a random variable, assuming each edge follows a Bernoulli distribution. A neural network computes the probability of an edge existing between node pairs. Computational complexity is reduced using k-nearest neighbor approximation, while entropy regularization prevents probabilities from converging to extreme values. The learned probabilistic adjacency matrix is symmetricized and fused with the original graph structure to form an enhanced probabilistic graph structure. Neighbor information is then aggregated layer-by-layer through a probability-weighted graph convolutional network. Hierarchical representations from the attribute autoencoder are injected at different levels to enhance discriminability. An inner-product decoder reconstructs the probabilistic graph structure to mitigate over-smoothing. Finally, the dual self-supervised

module computes soft assignments based on attribute representations to generate sharp target distributions. It simultaneously minimizes KL divergences between target distributions and attribute assignments, as well as between target distributions and graph assignments. This unifies attribute learning and probabilistic graph structure learning within an end-to-end framework. Through an alternating optimization strategy, network parameters and probabilistic graph structures are iteratively updated. Discriminative attribute representations guide the construction of more precise probabilistic graphs, while precise probability graphs further enhance node representations. This synergistic evolution ultimately yields high-quality clustering results that are robust to noise, compact within classes, and well-separated between classes [18].

The dual self-supervision mechanism proposed by our research fundamentally differs from mainstream graph contrastive learning methods (e.g., DGI, GRACE) in three key aspects. 1. Objective Function: Contrastive learning maximizes mutual information between views (requiring classifiers and negative samples), whereas our approach minimizes the KL divergence between attribute-view and graph-view clustering assignments, directly serving the clustering objective. Input format: Contrastive learning requires constructing multiple views (e.g., edge pruning, feature masking), introducing additional hyperparameters. Our approach only requires the original graph, with the two perspectives derived from distinct network architectures that are naturally complementary. Computational efficiency: Contrastive learning incurs significant negative sample overhead. Our KL divergence has O(NC) complexity (C being the number of clusters, typically ≤20), independent of negative samples.

The OPGC model formulates graph structure learning as a probabilistic problem, where each edge follows a Bernoulli distribution with an existence probability $P_{ij} = \chi\left(\mathrm{MLP}\left[z_i; z_j\right]\right)$, where χ represents the sigmoid function, and z_i and z_j denote node attribute representations. To ensure the edge probability constitutes a valid probabilistic constraint, the study introduces a normalization condition: $\sum_{j=1}^{n} P_{ij} = 1$. Once these constraints are satisfied, the adjacency matrix can be regarded as a valid probabilistic graph. The graph convolutional autoencoder module first uses adaptive graph learning to obtain the probabilistic graph adjacency matrix. To avoid trivial solutions, a regularization term needs to be introduced into the objective function to control the sparsity of the similarity vector, and the calculation is shown in Equation (9).

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\left\| g\left(x_i\right) - g\left(x_j\right) \right\|^2 \right) s_{ij} + \gamma_i s_{ij}^2 \tag{9}$$

In Equation (9), x_i and x_j indicate the original attribute features of nodes i and j, $g(\cdot)$ represents the mapping function, s_{ij} represents a non-negative vector, and γ_i represents the tradeoff parameter. The calculation of γ_i is shown in Equation (10).

$$\gamma_i = \frac{k}{2} d_{i,k+1} - \frac{1}{2} \sum_{j=1}^{k} d_{ij} \tag{10}$$

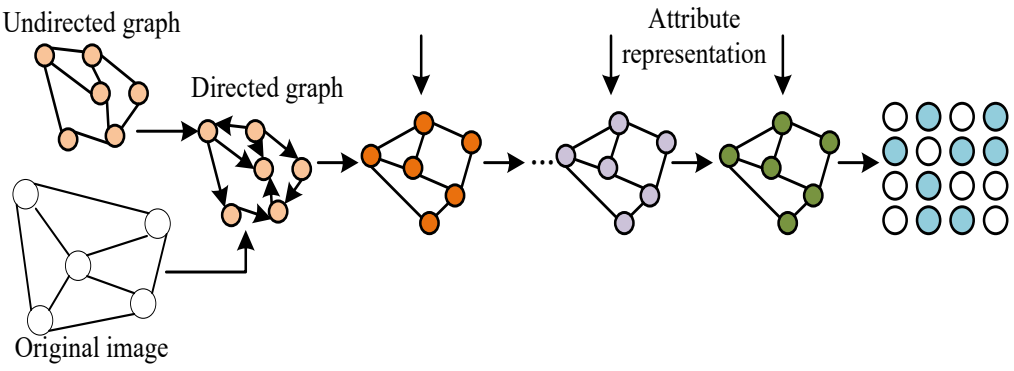

Figure 5. Specific structure of the graph convolutional encoder

In Equation (10), k denotes the total amount of neighboring nodes of the target node i, $d_{i,k+1}$ represents the $k+1$ the distance value, and d_{ij} means the distance between nodes i and j. The specific structure of the convolutional autoencoder is shown in Figure 5.

In Figure 5, the graph convolutional autoencoder first transforms the learned directed graph into an undirected graph and merges it with the original GS to form an enhanced GS, and then learns the corresponding structural embedding. At the same time, the attribute representation output by the attribute autoencoder module is introduced at different levels to improve the decision-making ability of the graph representation. The calculation of the enhanced GS is shown in Equation (11) [19].

$$\bar{S} = \mu M + (1-\mu) A \tag{11}$$

In Equation (11), $\bar{S}$ represents the enhanced GS, μ represents the GS enhancement weight parameter, M represents the undirected graph, and S represents the original GS. The GS reconstruction loss of the OPGC model can alleviate the oversmoothing problem caused by multi-layer graph convolution and restore a high-quality probabilistic GS, as shown in Equation (12).

$$L_{gsr} = -\sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij} \log \hat{s}_{ij} \tag{12}$$

In Equation (12), L_{gsr} represents the GS reconstruction loss of the OPGC model, and s_{ij} represents the true connection probability of nodes i and j. $\hat{s}_{ij}$ is obtained by calculating the normalized similarity of nodes i and j. The graph convolutional autoencoder and the adaptive graph learning module form a mutually reinforcing closed loop, using directed graphs to learn embedded representations, and the embedded representations are used as mapping functions to learn better directed graphs in the next round. Therefore, the study introduces dynamic sparsity adjustment to avoid premature fixation of the GS, and the calculation is shown in Equation (13).

$$k_{\max} = \min\left(|C_1|, |C_2|, ..., |C_m|\right) \tag{13}$$

In Equation (13), $k_{\max}$ represents the upper bound of the number of neighbors, which should ideally not exceed the size of the smallest cluster to ensure that connections only occur within nodes of the same type, and C_m denotes the total amount of samples in the cluster m. The SSC module solves the label missing problem in unsupervised learning through a dual self-supervised mechanism. First, based on the latent representation output by the attribute autoencoder, the similarity between each sample and each cluster center is calculated to obtain the initial SA probability distribution. The probability distribution is squared and normalized according to its category to generate a target probability distribution with higher confidence. The module constructs two self-supervised losses. The initial loss serves to enhance the discriminativeness of attribute features by minimizing the KL divergence between the target distribution and the SA of attribute representation. The second loss makes sure that graph representation learning and attribute features are the same by reducing the KL divergence between the same target distribution and the SA of graph embedding representation. This dual supervision mechanism organically combines attribute learning and GS learning in a unified framework, so that the two representations promote each other and evolve together during the training process, and finally generate a discriminative representation suitable for clustering tasks, thereby significantly improving clustering performance [20]. The total objective function of the OPGC model is calculated as shown in Equation (14).

$$\min L_z = L_{gsr} + \frac{w_1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\left\|z_i - z_j\right\|^2 s_{ij} + \gamma_i s_{ij}^2\right) + w_2 L_1 + w_3 L_2 \tag{14}$$

In Equation (14), L_z represents the total loss function, w_1, w_2 and w_3 all represent weight parameters, $\sum_{j=1}^{n}\left(\left\|z_i - z_j\right\|^2 s_{ij}\right)$ represents the graph Laplacian regularization term of graph nodes i and j, and L_1 and L_2 represent the dual SSC objective functions, respectively. The calculation is shown in Equation (15).

$$\begin{cases} L_1 = KL\left(P_H \| Q_H\right) \\ L_2 = KL\left(P_H \| Q_Z\right) \end{cases} \tag{15}$$

The model employs an alternating optimization strategy to iteratively update network parameters and adaptive graph matrices, enabling them to mutually reinforce each other. The discriminative representation can guide the construction of more accurate GSs.

4. Results and Analysis

4.1. Experimental Dataset

The dataset employed in this study consists of 9,298 16×16 grayscale handwritten digit images from the U.S. Postal Service (USPS), encompassing 10 categories (represented by digits 0–9)[1]. Each image was flattened into a 256-dimensional feature vector, comprising 10 classes. The Reuters News Dataset comprises 7,674 news documents across 8 categories, represented using TF-IDF features, and reduced to 2,000 dimensions via PCA[2]. During data preprocessing, TF-IDF features were first applied to remove stop words and low-frequency words from text data. Image pixel values were normalized to the [0,1] range, features underwent L2 regularization, citation network self-loops were removed, and the dataset was finally split into training/validation/test sets at a ratio of 70%/15%/15%, maintaining consistent category distribution.

4.2. Experimental analysis of adaptive graph learning model

The experimental hardware configuration consisted of an Intel Xeon Silver 4210R CPU, an NVIDIA GeForce RTX 3090 GPU (24GB VRAM), 128GB of DDR4 RAM, and a 1TB NVMe SSD. The software environment used Ubuntu 20.04 LTS as the operating system, PyTorch 1.11.0 + CUDA 11.3 as the deep learning framework, and Python 3.8 as the programming language. For model hyperparameters, the learning rate was determined using a grid search strategy within the range {0.1, 0.01, 0.001, 0.0005, 0.0001}. The NMI metric on the validation set served as the selection criterion, ultimately setting the learning rate to 0.001. The Adam optimizer was employed with 200 training epochs. The number of network layers was determined experimentally. The number of autoencoder layers was searched within {2, 3, 4, 5}, revealing that 4 layers (AGC) and 5 layers (OPGC) achieved a balance between representational capability and overfitting. The number of GCN layers was searched within {1,2,3,4}. Over-smoothing occurred beyond 3 layers (AGC) and 4 layers (OPGC). The number of neighbors was searched within {5,10,15,20} and selected based on validation set performance. The dataset used in the study included the US Postal Service (USPS) dataset with 9298 images across 10 categories and the Reuters News dataset with 7674 documents across 8 categories. The sensitivity analysis results for different parameters of the AGC model are shown in Figure 6. δ and λ represent the hyperparameters of the sparsity and low-rank constraints, respectively, and κ represents the balance parameter.

In Figure 6(a), as the sparse constraint parameters increased, the Clustering Accuracy (CACC) and NMI fluctuated, but remained around 90% and 68%, respectively. In Figure 6(b), the model's CACC and NMI remained around 91% and 70%, respectively, as the low-rank constraint parameters changed. In Figure 6(c), when the balance parameter changed, the model's NMI fluctuated significantly, but still remained around 3%. This denotes that the model has strong robustness to parameter fluctuations and can maintain model stability. Figure 7 showcases the comparison of the AGC model's CACC on different datasets.

[1] U.S. Postal Service (USPS) Handwritten Digit Dataset. Available at: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#usps. Contains 9,298 grayscale images of handwritten digits (0–9), each of size 16×16 pixels, flattened into 256-dimensional feature vectors. Accessed: 2024.

[2] Reuters-21578 Text Categorization Collection. Available at: https://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection. Contains 7,674 news documents across 8 categories. Features are represented using TF-IDF and reduced to 2,000 dimensions via principal component analysis (PCA). Accessed: 2024.

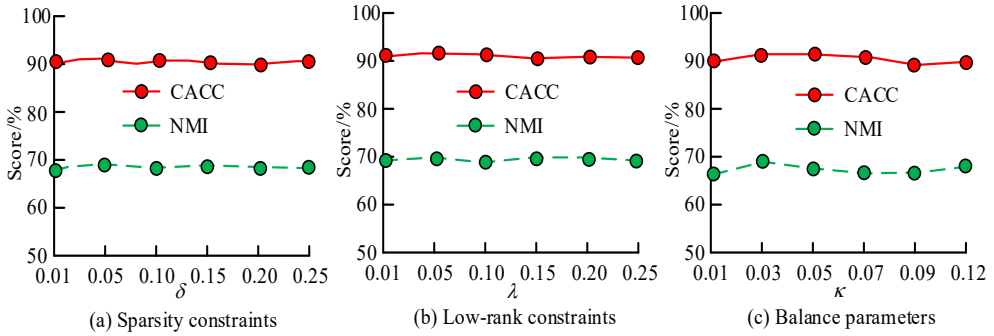

Figure 6. Sensitivity analysis results for different parameters of the AGC model

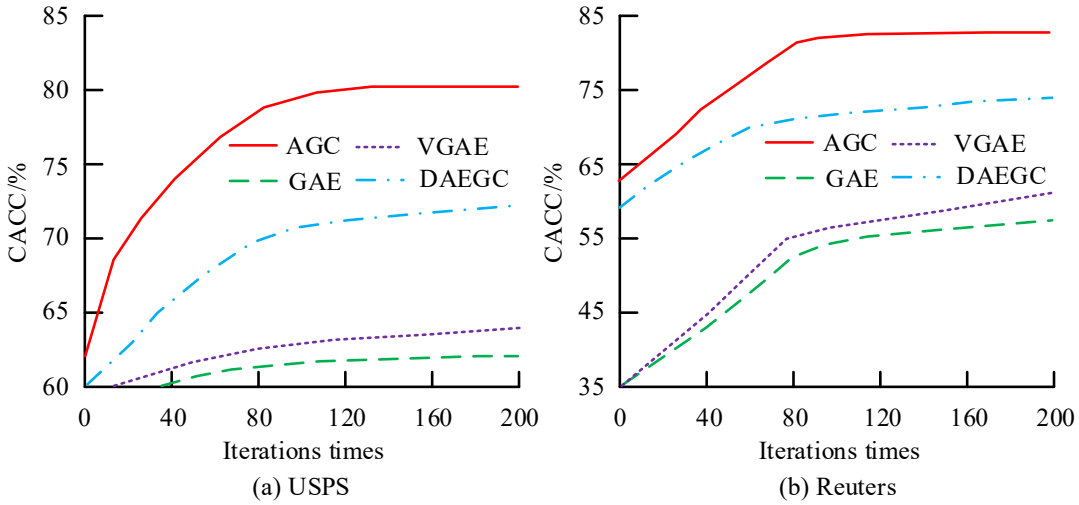

Figure 7. Comparison of CACC for the AGC model across different datasets

In Figure 7(a), the AGC model converged after 120 iterations, with a max CACC value of 80.2%, which was 17.9%, 7.8%, and 16.2% higher than GAE, Deep Adaptive Graph Clustering (DAEGC), and Variational Graph Autoencoder (VGAE), respectively. In Figure 7(b), the CACC values of the AGC and DAEGC models increased, while those of the GAE and VGAE models decreased significantly, indicating that they are not suitable for text data. The AGC model achieved a maximum CACC of 84.1%, which was 9.6% higher than the second-best performing DAEGC model, demonstrating that AGC has stronger adaptability and robustness when handling heterogeneous data. The stability comparison of the models under different noise levels is shown in Figure 8.

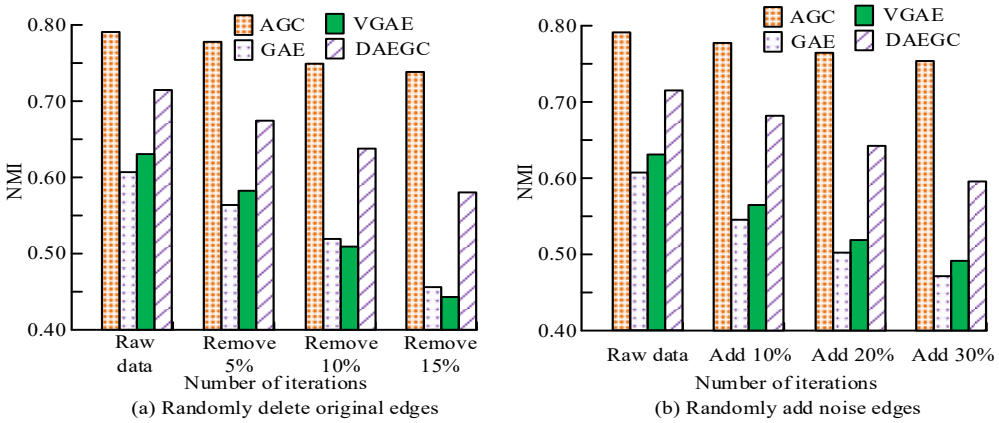

Figure 8. Comparison of model stability at different noise levels

In Figure 8(a), after removing the original edges, the NMI values of all models began to decline, with the VGAE model showing the largest decrease. The original NMI value was 0.786. After randomly removing 15%

of the original edges, the AGC model's NMI dropped to 0.742, representing an absolute decrease of 0.044 and a relative decrease of 5.6%. In comparison, the GAE model's NMI decreased from 0.623 to 0.525 (absolute decrease of 0.098, relative decrease of 15.7%), while the VGAE decreased from 0.658 to 0.509 (absolute decrease of 0.149, relative decrease of 22.6%), and the DAEGC decreased from 0.718 to 0.659 (absolute decrease of 0.059, relative decrease of 8.2%). The significantly smaller decrease in AGC compared to the comparison methods indicates its stronger structural robustness. In Figure 8(b), after adding 30% random noise edges, the NMI value of the AGC model decreased from 0.786 to 0.751, a decrease of 0.035. AGC had strong fault tolerance to GS perturbations and exhibited good structural robustness.

3.2. Experimental analysis of the optimal probabilistic graphical learning model

The hardware and software environments for the OPGC model were the same as those in Experiment 1. The autoencoder had a 5-layer symmetric structure, and the graph convolutional autoencoder had 4 layers. The initial number of neighbors was set to 10, the learning rate was 0.0005, the weight decay was 1e-4, and the amount of training iterations was 200. The performance comparison of the OPGC model on different datasets is denoted in Table 1. The comparison methods used include Graph Neural Network Clustering (GNNC), Graph Clustering Embedding (GCE), Graph Attention Networks (GATs), Spectral Clustering (SC), and Graph Clustering Regularization (GCR).

Table 1. Performance comparison of the OPGC model across different datasets

Model	CACC	NMI	ARI	F1
OPGC	0.852	0.812	0.785	0.839
GNNC	0.798	0.763	0.725	0.781
GCE	0.776	0.738	0.698	0.752
GATs	0.759	0.721	0.682	0.736
K-means	0.543	0.508	0.462	0.518
GCR	0.783	0.745	0.706	0.762
SC	0.632	0.595	0.549	0.608

In Table 1, the OPGC model ranked first in four metrics: ACC, NMI, Adjusted Rand Index (ARI), and F1 score. Its ACC reached 0.852, an improvement of approximately 6.8% compared to the second-place GNNC. Deep learning methods such as GNNC, GCE, and GATs significantly outperformed traditional methods. This indicates that the OPGC model achieves significant performance improvements in graph node clustering tasks through dynamic GS learning and a dual self-supervised mechanism. The ablation experiment results of the OPGC model are shown in Figure 9.

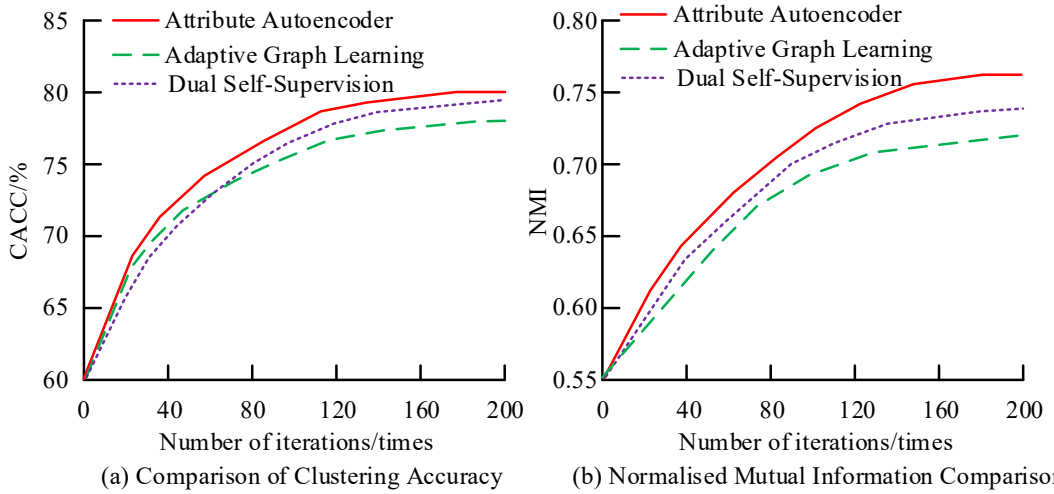

Figure 9. Ablation experiment results for the OPGC model

In Figure 9(a), all modules contributed positively to model performance, with adaptive graph learning being the most crucial for performance improvement. After removing the attribute encoder, adaptive graph learning, and dual self-supervision, the model's CACC value decreased by 0.050, 0.082, and 0.064, respectively. In Figure 9(b), after removing the adaptive graph learning module, the model's NMI value decreased to 0.725. The attribute encoder and dual self-supervision mechanism play a vital role in model representation learning and clustering optimization, validating the rationality of the OPGC model design and the synergistic effect among its modules. The clustering results of the OPGC model on different datasets are visualized in Figure 10.

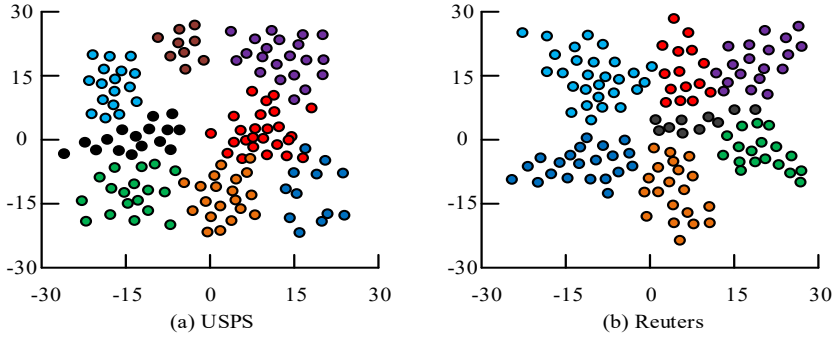

Figure 10. Visualization of clustering results from the OPGC model across different datasets

In Figures 10(a) and (b), only the primary categories are displayed to clearly present the clustering results. Before training, data across categories in both datasets were scattered and disorganized, lacking clear distribution boundaries. In Figure 10(c), the USPS dataset clusters into 8 primary categories. The OPGC model successfully aggregates different data types into distinct clusters, keeping them separated from others with well-defined boundaries. In Figure 10(d), data points from different categories are relatively dispersed, while those within the same cluster are concentrated. The model demonstrates strong clustering properties, effectively reducing confusion between different data types. The scalability and efficiency analysis of the OPGC model is shown in Table 2. To evaluate the significance of performance differences, paired two-tailed t-tests were conducted across 10 independent runs. For each metric, the p-value corresponds to the comparison between OPGC and the best-performing baseline (i.e., the baseline with the closest value to OPGC). The null hypothesis assumes no mean difference between the two models. A p-value < 0.05 indicates a statistically significant difference.

Table 2. Scalability and efficiency analysis of the OPGC model

Metric	OPGC	GNNC	GCE	GATs	GCR	p
Training Duration (minutes)	85.2 ± 3.1	92.7±4.2	88.5±3.8	104.3±5.1	79.8±2.9	< 0.01
Convergence Iterations	156±12	183±15	172±14	198±18	145±11	< 0.05
Processing Speed per Round (samples/second)	1,258±45	1,103±52	1,182±48	956±63	1,342±41	< 0.01
Peak Memory Usage (GB)	6.8±0.3	7.5±0.4	7.2±0.3	8.9±0.5	6.2±0.2	< 0.01
GPU Memory Occupancy (GB)	4.2±0.2	4.8±0.3	4.5±0.2	5.6±0.4	3.9±0.1	< 0.05
CPU Utilisation (%)	68.5±2.1	72.3±2.8	70.1±2.4	76.8±3.2	65.2±1.9	< 0.05
DBLP-10K ACC	0.846±0.008	0.791±0.012	0.769±0.014	0.752±0.016	0.778±0.011	< 0.01
DBLP-50K ACC	0.832±0.012	0.763±0.018	0.741±0.020	0.718±0.022	0.752±0.017	< 0.01
Amazon-100K ACC	0.815±0.015	0.728±0.023	0.705±0.025	0.682±0.028	0.735±0.021	< 0.01
Variance Across 10 Runs	0.00012	0.00028	0.00035	0.00042	0.00019	< 0.05
Maximum Performance Fluctuation	± 1.8%	± 3.2%	± 3.8%	± 4.5%	± 2.4%	< 0.01

Note: Values are mean ± standard deviation over 10 runs. The p-value is derived from a paired two-tailed t-test comparing OPGC against the best baseline for each metric (e.g., for training duration, the best baseline is GCR). A p-value < 0.05 indicates statistical significance.

In Table 2, the *p* value is calculated based on paired t-tests conducted across 10 independent runs. OPGC significantly outperformed GNNC and GATs in training time ($p < 0.01$) and converged 26.9% faster than GATs. OPGC used 23.6% less memory than GATs while maintaining 81.5% accuracy on a dataset with 100,000 nodes. The OPGC model demonstrate excellent computational efficiency and resource utilization while maintaining high performance. A comparison between the proposed method and the state-of-the-art methods is shown in Table 3.

Table 3. Comparison between the proposed method and the state-of-the-art methods

Method	Model Type	CACC	NMI	ARI
Spectral Clustering (SC)	Traditional	0.532	0.421	0.368
GCN	GNN	0.613	0.528	0.452
GAE	Graph Autoencoder	0.596	0.507	0.435
VGAE	Variational Graph Autoencoder	0.618	0.536	0.468
DGI	Contrastive Learning	0.682	0.593	0.521
DAEGC	Adaptive Graph Learning	0.724	0.648	0.576
GRACE	Contrastive Learning	0.713	0.635	0.562
IDGL	Dynamic Graph Learning	0.756	0.682	0.613
GCA	Contrastive Learning	0.738	0.661	0.594
AGC	Adaptive Graph Learning	0.802	0.742	0.689
OPGC	Probabilistic Graph Learning	0.851	0.814	0.785

In Table 3, the results of all the models being compared were all obtained by this study itself through experiments conducted under the same experimental conditions (including dataset division, evaluation metrics, and hardware environment). When compared against several state-of-the-art graph clustering methods, the CAGC and OPGC models demonstrate significant advantages across all evaluation metrics: CACC, NMI, and ARI. The CAGC and OPGC models achieved CACC values 0.046 and 0.095 higher than the second-best IDGL model, respectively. Their NMI scores reached 0.742 and 0.812, surpassing the standard GCN model by 0.214 and 0.286, respectively. The ablation experiment results for the OPGC model are shown in Table 4.

Table 4. Ablation Experiment Results for the OPGC Model

Model Variants	Dataset	CACC	NMI	ARI
OPGC (Complete)	USPS	0.852	0.812	0.785
	Reuters	0.841	0.798	0.769
V1: No Adaptation	USPS	0.791 (↓ 7.2%)	0.743 (↓ 8.5%)	0.702 (↓ 10.6%)
	Reuters	0.776 (↓ 7.7%)	0.725 (↓ 9.1%)	0.681 (↓ 11.4%)
V2: No Probability	USPS	0.803 (↓ 5.7%)	0.758 (↓ 6.6%)	0.721 (↓ 8.2%)
	Reuters	0.789 (↓ 6.2%)	0.741 (↓ 7.1%)	0.703 (↓ 8.6%)
V3: No Adaptation + No Probability	USPS	0.752 (↓ 11.7%)	0.698 (↓ 14.0%)	0.652 (↓ 16.9%)
	Reuters	0.731 (↓ 13.1%)	0.672 (↓ 15.8%)	0.623 (↓ 19.0%)
V4: No Dual Self-Supervision	USPS	0.778 (↓ 8.7%)	0.725 (↓ 10.7%)	0.683 (↓ 13.0%)
	Reuters	0.762 (↓ 9.4%)	0.708 (↓ 11.3%)	0.664 (↓ 13.7%)

In Table 4, V1 (non-adaptive) shows a 7.2% decrease in ACC and a 10.6% decrease in ARI on USPS compared to the full model. The larger ARI decline indicates that a fixed graph structure not only reduces accuracy but also severely compromises the overall consistency of the clustering partition. This demonstrates that even with probabilistic modeling and dual supervision, the model remains constrained by the initial graph quality if the graph structure remains fixed. V2 (no probability) shows a 5.7% decrease in ACC and an 8.2% decrease in ARI on USPS. Deterministic graph learning is prone to overfitting in noisy environments, blurring inter-cluster boundaries—a phenomenon corroborated by the significant ARI decline. V4 (no self-supervision) exhibits an 8.7% drop in ACC and a 9.9% drop in F1 on USPS. Comprehensive declines across all four metrics

demonstrate that without clustering-guided objectives, models optimized solely for reconstruction loss fail to learn clustering-friendly representations. V3 (simultaneously removing Adaptive + Probability) saw a 19.0% drop in ARI on Reuters—the largest decrease among all variants and significantly higher than the individual declines of V1 and V2. This proves that adaptive and probabilistic mechanisms are complementary, jointly forming the model's core advantage. Ablation experiment B demonstrates that the "adaptive" and "optimal probability" components in the title, along with the "dual self-supervision" framework, are indispensable. The four metrics collectively validate this conclusion from different perspectives. The model complexity experiments are compared as shown in Table 5.

Table 5 Comparison of Model Complexity Experiments

Scale	N	Simple implementation achieves $O(N^2)$	This implementation achieves O(Nk)
Small (Cora)	2.7×10^3	7.3×10^6	2.7×10^4
Medium (Amazon)	1.0×10^5	1.0×10^{10}	1.0×10^6
Large (OGB)	1.0×10^8	1.0×10^{16}	1.0×10^9

In Table 5, OPGC achieves a memory footprint of only 6.8GB on Amazon-100K ($N=10^5$) while maintaining an ACC of 81.5%, demonstrating the effectiveness of its linear-complexity design. The proposed sparse update strategy enables the model to handle graphs with tens of thousands to hundreds of thousands of nodes. However, for graphs exceeding millions of nodes (e.g., the OGB dataset, $N=10^6$~10^8), O(N log N) nearest neighbor retrieval may still become a bottleneck. Additionally, full-graph training requires loading all node representations at once, imposing limitations on GPU memory. For ultra-large-scale graphs, future work may explore the following directions: (1) Mini-batch sampling training: Partition the graph into subgraphs, training only one batch of nodes and their neighbors per iteration to avoid full-graph computation. (2) Distributed frameworks: Distribute k-NN search and graph convolutions across multiple machines and GPUs for parallel processing. (3) Quantized indexes: Employ approximation techniques like product quantization and HNSW to reduce retrieval memory and latency. (4) Streaming graph learning: Design incremental graph structure update mechanisms for dynamically growing graphs to avoid global recomputations.

4. Conclusions and Recommendations

This study addresses three challenges in graph clustering: sensitivity to initial graph quality, rigid graph structures, and the disconnect between graph construction and representation learning. It proposes a graph clustering framework based on Adaptive Graph Learning (AGC) and Optimal Probabilistic Graph Learning (OPGC). Experiments demonstrate that: (1) The adaptive graph learning mechanism effectively mitigates dependence on initial graph quality. The AGC model achieves an accuracy of 80.2% on the USPS dataset, surpassing DAEGC by 7.8 percentage points, demonstrating that dynamically optimizing graph structures facilitates superior node representations. (2) Probabilistic graph modeling enhances robustness against noise. Under extreme conditions with 15% edge deletions, OPGC's NMI drops only 5.6%, significantly lower than competing methods, demonstrating that explicitly modeling edge uncertainty helps models withstand structural perturbations. (3) The dual self-supervised strategy achieves synergistic optimization of attribute learning and structure learning. Ablation experiments reveal that removing either supervision signal degrades performance, confirming the mutual reinforcement of both representations is crucial for clustering tasks. (4) Computational efficiency analysis shows that OPGC converges 26.9% faster than GATs while maintaining high performance, consumes 23.6% less memory, and achieves 81.5% accuracy on datasets with 100,000 nodes, validating the method's scalability. However, this study has the following limitations: (1) Performance degrades when processing larger datasets. (2) Scalability bottlenecks: Training on a graph with 100,000 nodes takes 85.2 minutes; further optimization is required to handle graph structures with millions of nodes. (3) Requires tuning for specific datasets: The optimal learning rate is 0.001 for the USPS dataset, while it is 0.0005 for the Reuters dataset. (4) Limited scope of noise testing: Only random edge deletion/addition (15%/30%) was tested; attribute noise and adversarial attacks were not evaluated. Future work will focus on the following three areas: (1)

Scaling up to graphs with millions of nodes through mini-batch sampling or distributed training; (2) Improving robustness on larger and more diverse datasets; (3) Testing a wider range of noise types (attribute corruption, adversarial edges).

CRediT Author Contribution Statement

Lingguo Zou: Writing-original draft, Data curation, Formal analysis; Meihua Zhang: Writing-review & editing, Investigation, Methodology, Conceptualization.

References

[1] Shiping Wang, Jinbin Yang, Jie Yao, Yang Bai and William Zhu, "An overview of advanced deep graph node clustering", *IEEE Transactions on Computational Social Systems*, Print ISSN: 2329-924X, Online ISSN: 2329-924X, 22 February 2024, Vol. 11, No. 1, pp. 1302-1314, Published by IEEE, DOI: 10.1109/tcss.2023.3242145, Available: https://ieeexplore.ieee.org/document/10049408.

[2] Sandipan Choudhuri, Suli Adeniye and Arunabha Sen, "Distribution Alignment Using Complement Entropy Objective and Adaptive Consensus-Based Label Refinement for Partial Domain Adaptation", *Artificial Intelligence and Applications*, Print ISSN: 2817-8302, Online ISSN: 2817-8302, 5 January 2023, Vol. 1, No. 1, pp. 43-51, Published by Bon View Press, DOI: 10.47852/bonviewaia2202524, Available: https://bonvievpress.com/index.php/AIA/article/view/524.

[3] Lun Hu, Yue Yang, Zehai Tang, Yizhou He and Xin Luo, "FCAN-MOPSO: An improved fuzzy-based graph clustering algorithm for complex networks with multiobjective particle swarm optimization", *IEEE Transactions on Fuzzy Systems*, Print ISSN: 1063-6706, Online ISSN: 1941-0034, 20 March 2023, Vol. 31, No. 10, pp. 3470-3484, Published by IEEE, DOI: 10.1109/tfuzz.2023.3259726, Available: https://ieeexplore.ieee.org/document/10077417.

[4] Kamal Berahmand, Sogol Haghani, Mehrdad Rostami and Yuefeng Li, "A new attributed graph clustering by using label propagation in complex networks", *Journal of King Saud University-Computer and Information Sciences*, Print ISSN: 1319-1578, Online ISSN: 2213-1248, May 2022, Vol. 34, No. 5, pp. 1869-1883, Published by Elsevier, DOI: 10.1016/j.jksuci.2020.08.013, Available: https://www.sciencedirect.com/science/article/pii/S1319157820304511.

[5] Pablo Villanueva-Domingo and Francisco Villaescusa-Navarro, "Learning cosmology and clustering with cosmic graphs", *The Astrophysical Journal*, Print ISSN: 0004-637X, Online ISSN: 1538-4357, 4 October 2022, Vol. 937, No. 2, pp. 115-123, Published by IOP Publishing, DOI: 10.3847/1538-4357/ac8930, Available: https://iopscience.iop.org/article/10.3847/1538-4357/ac8930.

[6] Zongmo Huang, Yazhou Ren, Xiaorong Pu, Shudong Huang, Zenglin Xu *et al.*, "Self-supervised graph attention networks for deep weighted multi-view clustering", *in Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-23)*, 7–14 February 2023, Washington, DC, USA, Online ISBN: 978-1-57735-881-7, E-ISBN: 978-1-57735-880-0, pp. 7936–7943, Published by AAAI Press, DOI: 10.1609/aaai.v37i7.25960, Available: https://ojs.aaai.org/index.php/AAAI/article/view/25960.

[7] Chun Wang, Shirui Pan, Celina P. Yu, Ruiqi Hu, Guodong Long *et al.*, "Deep neighbor-aware embedding for node clustering in attributed graphs", *Pattern Recognition*, Print ISSN: 0031-3203, Online ISSN: 1873-5142, February 2022, Vol. 122, No. 5, pp. 108230-108239, Published by Elsevier, DOI: 10.1016/j.patcog.2021.108230, Available: https://www.sciencedirect.com/science/article/abs/pii/S0031320321005035.

[8] Anton Tsitsulin, John Palowitch, Bryan Perozzi and Emmanuel Müller, "Graph clustering with graph neural networks", *Journal of Machine Learning Research*, Print ISSN: 1532-4435, Online ISSN: 1533-7928, May 2023, Vol. 24, No. 127, pp. 1-21, Published by JLMR, Available: http://jmlr.org/papers/v24/20-998.html.

[9] Yue Yang, Xiaorui Su, Bowei Zhao, Guodong Li, Pengwei Hu *et al.*, "Fuzzy-based deep attributed graph clustering", *IEEE Transactions on Fuzzy Systems*, Print ISSN: 1063-6706, Online ISSN: 1941-0034, 1 September 2023, Vol. 32, No. 4, pp. 1951-1964, Published by IEEE, DOI: 10.1109/TFUZZ.2023.3338565, Available: https://ieeexplore.ieee.org/document/10339895.

[10] Nairouz Mrabah, Mohamed Bouguessa, Mohamed F. Touati and Riadh Ksantini, "Rethinking graph auto-encoder models for attributed graph clustering", *IEEE Transactions on Knowledge and Data Engineering*, Print ISSN: 1041-4347, Online ISSN: 1558-2191, 1 September 2023, Vol. 35, No. 9, pp. 9037-9053, Published by IEEE, DOI: 10.1109/TKDE.2022.3220948, Available: https://ieeexplore.ieee.org/document/9944925.

[11] Yue Liu, Wenxuan Tu, Sihang Zhou, Xinwang Liu, Linxuan Song *et al.*, "Deep Graph Clustering via Dual Correlation Reduction", *in Proceedings of the AAAI Conference on Artificial Intelligence 2022 (AAAI-22)*, 22 February – 1 March 2022, Virtual, Online ISBN: 978-1-57735-876-3, E-ISBN: 978-1-57735-875-6, pp. 7603–7611, Published by AAAI Press, DOI: 10.1609/aaai.v36i7.20726, Available: https://ojs.aaai.org/index.php/AAAI/article/view/20726.

[12] Wei Ju, Yiyang Gu, Binqi Chen, Gongbo Sun, Yifang Qin *et al.*, "GLCC: A General Framework for Graph-Level Clustering", *in Proceedings of the AAAI Conference on Artificial Intelligence 2023 (AAAI-23)*, 7–14 February 2023, Washington, DC, USA, Online ISBN: 978-1-57735-881-7, E-ISBN: 978-1-57735-880-0, pp. 4391–4399, Published by AAAI Press, DOI: 10.1609/aaai.v37i4.25559, Available: https://ojs.aaai.org/index.php/AAAI/article/view/25559.

[13] Aitao Yang, Min Li, Yao Ding, Xiongwu Xiao and Yujie He, "An efficient and lightweight spectral-spatial feature graph contrastive learning framework for hyperspectral image clustering", *IEEE Transactions on Geoscience and Remote Sensing*, Print ISSN: 0196-2892, Online ISSN: 1558-0644, 7 November 2024, Vol. 62, No. 5, pp. 1-14, Published by IEEE, DOI: 10.1109/TGRS.2024.3493096, Available: https://ieeexplore.ieee.org/document/10746460.

[14] Satu Sieranoja and Pasi Fränti, "Adapting k-means for graph clustering", *Knowledge and Information Systems*, Print ISSN: 0219-1377, Online ISSN: 0219-3116, January 2022, Vol. 64, No. 1, pp. 115-142, Springer Nature, DOI: 10.1007/s10115-021-01623-y, Available: https://link.springer.com/article/10.1007/s10115-021-01623-y.

[15] Hong Jun Li and Yu Feng, "Overlapping graph clustering in attributed networks via generalized cluster potential game", *ACM Transactions on Knowledge Discovery from Data*, Print ISSN: 1556-4681, Online ISSN: 1556-472X, January 2024, Vol. 18, No. 1, pp. 1-26, ACM, DOI: 10.1145/3597436, Available: https://dl.acm.org/doi/abs/10.1145/3597436.

[16] Yue Liu, Xihong Yang, Sihang Zhou, Xinwang Liu, Zhen Wang *et al.*, "Hard Sample Aware Network for Contrastive Deep Graph Clustering", *in Proceedings of the AAAI Conference on Artificial Intelligence 2023 (AAAI-23)*, 7–14 February 2023, Washington, DC, USA, Online ISBN: 978-1-57735-881-7, E-ISBN: 978-1-57735-880-0, pp. 8914–8922, Published by AAAI Press, DOI: 10.1609/aaai.v37i7.26071, Available: https://ojs.aaai.org/index.php/AAAI/article/view/26071.

[17] Nairouz Mrabah, Mohamed Bouguessa, Mohamed F. Touati and Riadh Ksantini, "Rethinking graph auto-encoder models for attributed graph clustering", *IEEE Transactions on Knowledge and Data Engineering*, Print ISSN: 1041-4347, Online ISSN: 1558-2191, 10 November 2022, Vol. 35, No. 9, pp. 9037-9053, Published by IEEE, DOI: 10.1109/TKDE.2022.3220948, Available: https://ieeexplore.ieee.org/document/9944925.

[18] Namkyeong Lee, Junhyun Lee and Chanyoung Park, "Augmentation-Free Self-Supervised Learning on Graphs", *in Proceedings of the AAAI Conference on Artificial Intelligence 2022 (AAAI-22)*, 22 February – 1 March 2022, Virtual, Online ISBN: 978-1-57735-876-3, E-ISBN: 978-1-57735-875-6, pp. 7372–7380, Published by AAAI Press, DOI: 10.1609/aaai.v36i7.20700, Available: https://ojs.aaai.org/index.php/AAAI/article/view/20700.

[19] Yan Chen, Tian Shu, Xiaokang Zhou, Xuzhe Zheng, Akira Kawai *et al.*, "Graph attention network with spatial-temporal clustering for traffic flow forecasting in intelligent transportation system", *IEEE Transactions on Intelligent Transportation Systems*, Print ISSN: 1524-9050, Online ISSN: 1558-0016, August 2023, Vol. 24, No. 8, pp. 8727-8737, Published by IEEE, DOI: 10.1109/tits.2022.3208952, Available: https://ieeexplore.ieee.org/document/9912369.

[20] Nadheeha Bandara, Thivya Kandappu, Anindya Sen, Ishan Gokarn and Archan Misra, "EyeGraph: modularity-aware spatio temporal graph clustering for continuous event-based eye tracking", *Advances in Neural Information Processing Systems*, Print ISSN: 1049-5258, Online ISSN: 1049-5258, December 2024, Vol. 37, No. 2, pp. 120366-120380, Published by Neural Information Processing Systems Foundation, DOI: 10.52202/079017-3825, Available: https://doi.org/10.52202/079017-3825.