The Journal of Supercomputing (2024) 80:25933–25973
https://doi.org/10.1007/s11227-024-06436-8

Community detection in attributed social networks using deep learning

Omid Rashnodi[1] · Maryam Rastegarpour[2] · Parham Moradi[3] · Azadeh Zamanifar[1]

Accepted: 6 August 2024 / Published online: 16 August 2024

Abstract
Existing methods for detecting communities in attributed social networks often rely solely on network topology, which leads to suboptimal accuracy in community detection, inefficient use of available data, and increased time required for identifying groups. This paper introduces the Dual Embedding-based Graph Convolution Network (DEGCN) to address these challenges. This new method uses graph embedding techniques in a new deep learning framework to improve accuracy and speed up community detection by combining the nodes' content with the network's topology. Initially, we compute the modularity and Markov matrices of the input graph. Each matrix is then processed through a graph embedding network with at least two layers to produce a condensed graph representation. As a result, a multilayer perceptron neural network classifies each node's community based on these generated embeddings. We tested the suggested method on three standard datasets: Cora, CiteSeer, and PubMed. Then, we compared the outcomes to many basic and advanced approaches using five important metrics: F1-score, adjusted rand index (ARI), normalized mutual information (NMI), and accuracy. The findings demonstrate that the DEGCN accurately captures community structure, achieves superior precision, and has higher ARI, NMI, and F1 scores, significantly outperforming existing algorithms for identifying community structures in medium-scale networks.

Keywords Community detection · Attributed social networks · Graph convolution networks · Deep learning · Embeddings

1 Introduction

The exploration of community structures within networks has evolved significantly since early sociological studies, becoming a crucial research domain that utilizes sophisticated mathematical tools for large-scale data analysis. Notably,

Extended author information available on the last page of the article since Girvan and Newman's pioneering efforts in 2002, identifying and understanding these structures have become essential for comprehending the composition and functionality of various networks, impacting fields ranging from epidemiology to marketing.

Even though topological, content-based, and graph-theoretical approaches have progressed with community detection, there are still many problems with the current methods, especially with the quality of the vector representations for network nodes. Current methods often produce representations that fail to capture the structural and contextual information of the nodes fully. This means that they don't work well enough for tasks like clustering or classification, and they can't keep up with the rising quality standards as networks get more extensive and complicated.

This paper addresses the limitations of conventional community detection methods, mainly when applied to large-scale or high-dimensional networks constrained by computational capacity and data volume. These constraints significantly hamper the efficacy of traditional methods for analyzing complex, modern relational data. As a result, to embed graphs effectively, this study employs graph neural networks (GNNs), a specialized branch of deep learning tailored to graph data. This method simplifies networks by reducing their dimensions and enhancing the representation of their nodes, thereby expediting the community detection process. This study also combines the modularity matrix with the Markov matrix to improve the accuracy of community detection. This makes these methods more useful in complex network structures because they are more efficient and accurate.

The contributions and innovations of this study are summarized as follows: (a) we introduce a layout method utilizing graph convolution networks to reduce representation dimensions and decrease algorithm implementation time; (b) we implement a novel graph layout technique that preserves the structural and content features of nodes in the primal graph, enhancing node similarities; (c) we enhance the efficiency of community detection by employing objective functions that reflect the structural and content features of the network; and (d) we address the challenges posed by non-community-centric GNNs by combining modularity and normalized cuts, thereby improving neural network interpretability.

Community detection is acknowledged as a complex, NP-hard problem encompassing a broad spectrum of challenges. This paper tackles these challenges by focusing on computational complexity and detection accuracy within attributed social networks. By utilizing GNNs, we aim to overcome these issues through innovative approaches that embed networks and enhance graph representation learning. These strategies lead to a comprehensive solution for community detection that offers several benefits:

- Innovative use of GNNs: employing GNNs within deep learning to embed networks and learn graph representations addresses computational complexity, scalability, and detection accuracy challenges.
- Decreasing computational complexity: GNNs simplify computational demands by efficiently representing nodes and their connections in a reduced-dimensional space, achieved by performing local operations on nodes and their neighbors.

- Enhancing scalability: GNNs manage local and global network information within a unified framework, effectively handling medium-scale networks with hundreds of nodes and edges.
- Enhancing detection accuracy: deep learning and graph-based neural networks identify intricate structures and patterns in network data, thereby increasing the precision of community detection.
- Effective learning of graph representations: GNNs create network representations incorporating topological structures and node features, which are essential for accurately detecting communities.

We structure the remainder of this paper as follows: Sect. 2 surveys the existing literature on graph convolutional networks and dual embedding techniques, outlining fundamental advances and pinpointing gaps that our study addresses. Section 3 introduces the necessary concepts and notations, laying the groundwork for understanding the methodologies discussed subsequently. Section 4 provides a detailed description of the proposed algorithm DEGCN and presents its pseudocode. Section 5 is comprehensive and divided into three distinct parts: Sect. 5.1 offers an in-depth description of the datasets used for testing, outlines the evaluation metrics to assess performance, and details the chosen parameters and the experimental setup. Section 5.2 explains the ablation studies. Section 5.3 demonstrate graph visualization. Finally, Sect. 6 addresses the conclusion and future work.

2 Literature review

With recent developments in information technology and the digital world, complex network theory can be used in many fields, such as social networks, biological networks, and internet networks. Community detection is one of the critical issues in studies on complex networks seeking to discover the structural properties of networks. The communities are created in the network by a group of nodes with more incredible internal relationships and fewer relationships with other nodes. Early community detection methods are mainly based on the network's typological features, and many approaches have been proposed based on different criteria for similarity and closeness among groups. Before developing deep learning methods, community detection methods were divided into two general groups: hierarchical methods and partitioning methods. Hierarchical methods start from a partition where each node is considered a cluster or from a partition where all nodes are in a similar community. Then, clusters are continuously combined and divided using quality measurement to form the hierarchy of clusters. Although hierarchical methods do not need to know the number of communities as a background, they require a criterion to identify relevant partitions.

On the other hand, the partitioning methods identify clusters based on the iteration of member allocation. These methods evaluate the quality of the resulting partitions by optimizing one or more objective functions. Some of the partitioning techniques include finding the maximum number of cliques on a graph [1], modularity maximization [2], matrix decomposition [3], seed expansion [4], linear sparse coding [5], sparse linear coding [5], and evolutionary algorithms [1]. Both hierarchical and partitioning methods involve high computational costs. Thus, these methods cannot be efficiently implemented in large-scale networks. In other words, they cannot find a desirable solution over a reasonable period. More flexible local methods have been proposed for separate and overlapping detection communities [6]. For example, methods based on label propagation use the regional expansion of node labels to identify communities in linear time [7].

Deep learning (DL) techniques are increasingly utilized across various fields, including computer and social sciences, economics, agriculture, healthcare, and medicine [8]. Network representation learning (NRL) transforms complex network structure data into a manageable, low-dimensional space within this broad application spectrum. These methods include learning network representations [9], Network embedding [10], and graph embedding [11], all aimed at preserving the network's typological structure, vertex content, and auxiliary data.

By enabling robust graph data representation, these innovative learning approaches have revolutionized the construction of complex classification, clustering, and prediction models. This approach allows for executing analytical tasks using simpler, traditional models. NRL techniques aim to get representations of network vertices in fewer dimensions while keeping important network typological and content properties[9]. Machine learning tasks, such as node classification and link prediction, utilize the derived representations as vector inputs. This advancement has spurred the development of sophisticated and effective NRL methodologies tailored for complex networks [10]. Graph representation learning methods are broadly categorized into three distinct groups: probabilistic models, deep learning-based algorithms, and matrix decomposition algorithms. The subsequent discussion will elucidate these three different models:

- *Probabilistic Models*: Techniques such as LINE [12] and Node2vec[13] focus on extracting diverse graph patterns to learn embeddings. Node2vec maps nodes into a vector space, enhancing link prediction and node classification capabilities. Large-scale use of LINE is well known. It uses edge sampling to get around the problems that stochastic gradient descent usually has. This improves graph embedding processes without reducing efficiency.
- *Deep Learning-Based Algorithms*: DeepWalk [14] exemplifies the application of deep learning in graph theory. DeepWalk is very good at encoding complete structural data, even when some is lost. It does this by using the local structural information of vertices and adding the Skip-Gram model to the framework of random walks. This method has proved particularly effective in social networks when performing multilabel classification. Deep learning models take advantage of the nonlinear dynamics of extensive, complicated networks. They collect many types of relational data, like details about nodes, neighbors, edges, subgraphs, and community properties. These models are ideally suited for handling sparse networks and excel in unsupervised learning scenarios. Algorithms such as DNGR[15], SNDE [15], and ANRL [15] employ deep autoencoder models for high-dimensional data representation. At the same time, other end-to-end network-based methods, such as SNE[16] and

DeepGL[17], combine structural and attribute data to enrich graph representation learning. A single-layer autoencoder makes the MGAE algorithm [18] more accessible for clustering tasks, and HNE[19] combines deep autoencoder neural networks with convolutional networks to process adjacent vectors and images.

- *Matrix decomposition algorithms*: This group includes methods such as M-NMF [20] and TADW [21], which focus on decomposing matrices to learn node representations effectively. These techniques are pivotal in disentangling complex network structures, facilitating more profound insights into network dynamics and interactions.

Collectively, these methods provide a robust framework for handling and analyzing complex networks across various domains, supporting a wide range of applications from theoretical research to practical, real-world problem-solving.

Wang et al. [22] obtained deep representations with a graph autoencoder and implemented a spectral clustering algorithm for representations of graph clustering. Similarly, He et al. [23] proposed a nonlinear restructuring method for a modularity matrix based on deep neural networks; they then extended the technique into a semi-supervised community detection algorithm with a combination of paired limitations in graph nodes. Both problems are challenging due to high computational costs and the need to adjust many parameters; for example, the number of clusters is generally unknown in many large and heterogeneous networks worldwide. Recently, graph neural network (GNN)-based methods, including graph convolutional networks (GCNs), have been introduced to graphs to solve the issue of community detection [24, 25]. GCNs integrate the information of neighboring nodes in the deep convolutional layers of graphs. GCNs use convolutional operations such as convolutional neural networks to extract properties from network typology and node properties to represent complex properties of the community [26].

Since GCN was not initially employed for community detection, it does not focus on community structure when learning node embedding, and there are no limitations on the structural adaptations between communities and nodes. For this purpose, Jin et al.[27] proposed a semi-supervised GCN community detection model named MRFasGCN, which was obtained by combining a GCN with the Markov random fields statistical model to detect communities. The Markov random field has been extended as a new convolutional layer that makes the MRFasGCN and monitors the gross results of the GCN.

Sun et al. [28] proposed a framework to learn network embedding for cluster nodes in attributed graphs. Specifically, this framework simultaneously learns representations based on graphs and cluster-oriented representations. This framework consists of three modules: a graph autoencoder module, a soft modularity maximization module, and a self-clustering module. The graph autoencoder module learns node embedding based on typological structure and node properties.

Jin et al. [29] also proposed an unsupervised model to detect communities through GCN embedding. In this model, the GCN has been used as the main structure of the encoder to match two information sources, namely typology and property, where a dual encoder has been used to extract two different embeddings.

Luo et al. [30] proposed a deep learning model to find communities and structural holes simultaneously. The main framework used in this model is the GCN-based encoder. The GCN is an effective method for combining network typology and node properties in community detection. However, there are two problems with using GCNs in community detection: (1) The GCN learns the representation of hidden layers through encoding typological features and node properties, and it does not consider community features. Thus, the embedding obtained from a GCN will not be community-based. (2) This is a semi-supervised model rather than an unsupervised model.

Wang et al. [31, 32] proposed a new nonnegative matrix decomposition model along with two sets of parameters, a community membership matrix and a community characteristic matrix. Additionally, several efficient updating rules were proposed for evaluating the parameters while guaranteeing convergence. Using node attributes improves community detection and provides a semantic interpretation for the communities of the obtained communities.

Additionally, attempts have been made to develop semi-supervised methods to detect communities based on learning network representations where data labels have been combined via graph-based regulation to identify unlabeled nodes. Young et al. [33] used node representation to predict the background of a network and employed node labels to create various transfer learning and inductive learning methods. Recently, graph convolutional networks have been introduced for network analysis. Methods based on GCNs contribute to network typology and attribute data, unlike most semi-supervised methods that focus on network structure; however, these methods depend on many node labels to identify unlabeled nodes. Sun et al. proposed a network embedding framework based on a graph convolutional autoencoder for cluster nodes. In addition, some unsupervised approaches have been proposed recently.

In [34], a model supervised in the CNN framework was proposed for typological defect networks, where the model has two CNN layers with max-pooling operators to represent the network and a wholly connected DNN layer to detect the community. In these models, convolutional layers show the local attributes of each node from different perspectives. The experiments conducted on this model in TINs, with 10% labeled nodes and 90% unlabeled nodes, achieved an 80% accuracy of community detection, which shows that high-order neighbor representation can improve the accuracy of community detection.

In [35], a supervised community detection model named the linear graph neural network (LGNN) has been proposed to improve SBM efficiency in community detection and reduce computational costs. A linear graph neural network (LGNN) learns the represented attributes of nodes in direct networks by combining the operator without backoff and rules for messaging.

In [36], CommDGI optimizes graph representation and clustering jointly through mutual data on nodes and communities and maximizes graph modularity. This method uses *k*-means to cluster nodes by targeting cluster centers.

Spectral GCNs show all the attributes hidden from a node's neighborhood. The characteristics of neighboring nodes converge to the same values by frequently implementing Laplacian operations in the deep layers of the GCN. However, these models lead to over-smoothing in detecting communities. Graph convolutional ladder-shaped networks have been developed to alleviate such negative impacts as a new GCN architecture for unsupervised community detection based on U-Net in the CNN field [37].

Since different types of links are considered simple edges, GCNs show each link separately and add them together, leading to representation redundancy. IPGDN [38] distinguishes neighborhoods into different sections and automatically discovers the independent hidden attributes of a graph, such that it decreases the difficulty of detecting communities. The IPGDN is supported by the Hilbert–Schmidt independence criterion in neighborhood routing. Adaptive graph convolution has been introduced to detect communities in attributed graphs. This type of graph depends on structural data and representation features for detecting communities through GCNs, where neighboring nodes and nodes with similar attributes are categorized into the same cluster community. Therefore, in this method, two graph signals are multiplied, and high-frequency noise needs to be filtered. For this purpose, adaptive graph convolution involves the design of a low-pass graph filter with a frequency response function.

In [39], a potent method utilizing Cayley polynomials was proposed to achieve high-order approximation within the spectral convolutional architecture of graph neural networks. While only a handful of studies have explored GCN filters, CayleyNets are notable for utilizing low-pass filters that harness extensive community data for identification purposes.

In [40], they have addressed the limitations of graph convolutional neural networks in processing complex relational graphs, such as excessive smoothing during node classification. The newly introduced SM-GCN model aims to increase node categorization accuracy by reducing reliance on individual features and integrating scattering embeddings to counteract the over-smoothing effect.

In [41], researchers presented a novel framework called the graph convolutional fusion model (GCFM) for community detection in multiplex networks. The primary objective of this model is to enhance the accuracy of community detection in networks that consist of multiple layers, where each layer represents a distinct type of relationship among the same set of nodes. The GCFM uses a graph convolutional autoencoder for each layer to do this. This model lets it record and encode the structural features within each layer while considering the nearby nodes.

In [42], the TANMF algorithm is designed to identify dynamic modules within cancer temporal-attributed networks, integrating genomic data and temporal networks to transform. The experimental results demonstrated that TANMF is more accurate than existing methods, enriches identified modules with known pathways, and correlates with patient survival.

In [43], the jLDEC algorithm for identifying dynamic communities in temporal networks integrates graph representation learning, community detection, and the dynamics of network edges into a cohesive framework to enhance the accuracy of community detection. The results demonstrate superior performance over traditional methods, especially in accurately characterizing the dynamics of community structures in temporal networks.

In [44], the NE2NMF algorithm detects dynamic communities within networks by integrating network embedding and nonnegative matrix factorization. It boosts accuracy via a third-order smoothness strategy, which considers previous, current, and subsequent network snapshots, thereby enhancing the characterization of community dynamics. The experimental results confirm that NE2NMF outperforms traditional methods in terms of both accuracy and robustness.

In [45], the jLMDC algorithm was introduced for dynamic community detection in temporal networks, emphasizing the joint learning of feature extraction and clustering. This approach significantly boosts the accuracy and efficiency of detecting dynamic communities by integrating these processes into a unified framework. Compared to existing methods, this approach demonstrates substantial improvements in accuracy and reduced computation time, underscoring its ability to manage large-scale networks and complex community dynamics.

In [46], the DANMF-MRL method introduces a new technique known as the Deep Autoencoder-like NMF for MRL. This approach utilizes a deep encoding procedure to generate a representation matrix, which is then decoded to reconstruct the original data. By employing a framework based on DANMF, we effectively address the challenges of consistency and complementarity in multi-view data, significantly enhancing the depth and comprehensiveness of the data representation.

In [47], a nonnegative matrix factorization-based MRL framework was proposed to consider two essential components jointly. Specifically, the exclusivity term is designed to leverage diverse intra-view information, while the consistency term ensures a unified representation across multiple views. Additionally, a local manifold component is integrated to maintain the local geometric structure of the data. Finally, they introduced a multiplicative-based alternating optimization algorithm to address this problem, complete with proof of convergence.

In [48], a hypergraph regularized diverse deep matrix factorization (HDDMF) model is introduced for multi-view data representation. This model combines multi-view diversity with high-order manifold analysis within a multilayer factorization framework. A newly designed diversity enhancement term exploits the structural complementarity across different data views. Hypergraph regularization is also employed to preserve the high-order geometric structures within each view. Furthermore, an efficient iterative optimization algorithm is developed to implement this model, accompanied by a theoretical analysis of its convergence.

In [49], several domain adaptation and transfer learning methods are used for social network analysis. These methods are applicable when the source domain has some labeled data but the target domain lacks labels. The goal is to transfer knowledge from the source domain to the target using a small training set based on deep learning.

Review studies in this field indicate that graph embedding methods can significantly enhance efficiency and reduce the time required for community detection in social networks. This article uses two parallel graph convolutional networks (GCNs), a deep learning-based embedding method for network representation learning. On the other hand, one big problem with GCNs is that they aren't naturally community-oriented. This means that the node representations made by these methods might not be accurate enough, which could make it harder to find communities. To address this issue and improve the interpretability of the representations, we first use the *k*-core algorithm to filter the graph and remove less significant nodes, thereby reducing the graph's size and making the communities within the graph more distinct. Subsequently, one GCN embeds the modularity matrix (representing the graph's structure), and the other embeds the Markov matrix (representing node content). We then average these two embedded representations to create the final representation, which is more meaningful and has smaller dimensions than the initial state for community detection.

3 Preliminaries and notation

This section briefly introduces preliminary knowledge, including basic signs and problem statements.

3.1 Attributed graph

Suppose that $G = (V, E, A, X)$ is an attributed network where V is a set of vertices $\left\{v_1, v_2, \ldots, v_n\right\}, E$ is a set of edges between nodes, A is the adjacency matrix, and X is the attribute matrix where an element X_{ip} represents the value of the p-th attribute for the vertexv_i. In adjacency matrix A, if there is an edge between the two vertices of v_i and v_j then$a_{\mathrm{ij}} > 0$. For weightless networks, if there is an edge, $a_{\mathrm{ij}} = 1$; otherwise, $a_{\mathrm{ij}} = 0$. If the network is not direct, $a_{\mathrm{ij}} = a_{\mathrm{ji}}$ also holds [50].

3.2 Community and community detection

Consider that we have the community set $C = \left\{C_1, C_2, \ldots, C_r\right\}$. Each community is a network partition with regional structures and shared cluster attributes. The node v_i that is clustered in the community C_i It should meet the condition that the internal degree of every node is greater than its external degree. In this paper, community detection is considered in the attributed graph. The graph has G attributes and the number of r communities. This paper aims to find the function $f : v \rightarrow \{1,2,3, \ldots, r\}$ such that r is true for all $f\left(v_i\right) = r$ nodes of the r community. Function partitions should follow the following principles: (1) Nodes of a group are connected, while the nodes are not connected in different groups. (2) Nodes in the same community tend to have similar attribute values, while those from different communities may vary relatively, even if they are neighbors at the graph level. (3) The function can adequately maintain the attributed graph's node attributes and structural information. Finally, we can find the groups separate from the nodes and their inductive subnodes, i.e., communities.

3.3 Decomposition *k*-core

Assume a graph G = (V, E) of $|V| = n$ vertices and $|E| = e$ edges; a *k*-core is defined as follows: A subgraph $H = (C, E|C)$ induced by the set $C \subseteq V$ is a *k*-core or a core of order k iff $\forall\ v \in C$: degree $H(v) \geq k$, and H is the maximum subgraph with this property. Therefore, a k-core of G can be obtained by recursively removing all the vertices of degrees less than k until all vertices in the remaining graph have at least degree k.

3.4 Modularity and normalization cut

Assume that network $G=(A, S)$ is undirected and attributed to n nodes, where $A = [a_{\mathrm{ij}}] \in R^{n*n}$ is the adjacency matrix. In this matrix $a_{\mathrm{ij}} = 1$ if there is an edge between nodes i and j; otherwise,$a_{\mathrm{ij}} = 0$. Here, $\beta_i = \sum_j a_{\mathrm{ij}}$ is the degree of node i, and $m = \frac{1}{2} \sum_i \beta_i$ is the total number of network edges. $S = [s_{\mathrm{ij}}] \in R^{n*n}$ is a similarity matrix in which s_{ij} is the cosine similarity value between the corresponding content vectors of nodes i and j. According to these explanations, the normalized cut and modularity models are defined as follows:

3.4.1 Modularity model

The modularity function Q was introduced by Newman-Girvan in [51]. This function is by far the most well-known quality function for community detection. Therefore, Q modularity optimization has become one of the leading community detection methods. Equation (1) defines this function for two communities:

$$\emptyset = \frac{1}{4\mathrm{m}} \sum_{\mathrm{ij}} \left(a_{ij} - \frac{\beta_i \beta_j}{2\mathrm{m}} \right) \left(\psi_i \psi_j \right) \tag{1}$$

where ψ_i is equal to 1 (or -1) if node v_i belongs to community 1 (or 2). Modularity can be easily optimized using specific vectors and values by defining a modularity matrix, as shown in Eq. (2):

$$B = \left[b_{\mathrm{ij}} \right] \in R^{n*n},\ \text{with entries} b_{\mathrm{ij}} = a_{\mathrm{ij}} - \frac{\beta_i \beta_j}{2\mathrm{m}} \tag{2}$$

Therefore, the modularity $\emptyset$ can be rewritten as Eq. (3):

$$\emptyset = \frac{1}{4\mathrm{m}} \psi^T B \psi \tag{3}$$

where $\psi = [\psi_i] \in \{-1,1\}^n$ represents membership in a community node. However, maximizing modularity is an NP-hard problem. By simplifying the problem and allowing variables ψ_i to take any integer value, the problem can be easily solved as Eq. (4):

$$\max \emptyset = \max \mathrm{Tr}\left(\Psi^T B \Psi \right) \tag{4}$$

where $\Psi = [\psi_{ij}] \in R^{n*p}$ is the matrix that hints at membership in the community and Tr (0) is the trace function. The solution is to obtain p of the most significant specific vector of modularity matrix B. In addition, the solution space allows Ψ

reconstruction of network topology from a community structure viewpoint. Therefore, any row of the Ψ matrix can be assumed to be a good representation of the corresponding node in the hidden space to detect the community.

3.4.2 Normalize cut model

This model yields the ratio of the number of external edges to the number of internal edges. To calculate a normalized cut, the cut between clusters *A* and *B*, denoted as Cut (*A*, *B*), is the total number of edges with only one node. The volume of cluster *A*, denoted as Vol (*A*), is the sum of the node degrees in cluster *A* [52]. These are computed using Eqs. (5) and (6):

$$\mathrm{cut}(A,B)=\sum_{i\in A,j\in B} w_{\mathrm{ij}} \tag{5}$$

$$\mathrm{Vol}(A)=\sum_{i\in A} k_i \tag{6}$$

Given Eqs. (5) and (6), the objective function of the normalized cut for two clusters, *A* and *B*, is Eqs. (7) or (8) when there are *k* clusters C_1, C_2 … C_k.

$$\mathrm{Ncut}(A,B)=\frac{\mathrm{cut}(A,B)}{\mathrm{vol}(A)}+\frac{\mathrm{cut}(A,B)}{\mathrm{vol}(B)} \tag{7}$$

$$\mathrm{Ncut}\left(C_1,C_2,\dots,C_k\right)=\sum_{t=1}^{k}\frac{\mathrm{link}\left(C_t,\overline{C}_t\right)}{\mathrm{vol}\left(C_t\right)} \tag{8}$$

where $\mathrm{link}\left(C_t,\overline{C}_t\right)=\frac{1}{2}\sum_{i\in C_t,j\in\overline{C_t}} S_{\mathrm{ij}}$ is the total connection from nodes in $\mathrm{C_t}$ to all nodes in $\overline{C}_t$ (not in C_t) and $\mathrm{vol}\left(C_t\right)=\sum_{i\in C_t} d_i$ is the total internal connection in C_t.

To achieve the minimum objective function, the normalized cut is wrapped in an optimization problem as per Eq. (9), where *L* is the Laplacian graph matrix of similarity and its normalized form $D^{-1}L=D^{-1}(D-S)=I-D^{-1}S$ is the identity matrix (*I*). Equation (10) is known as the Markov matrix:

$$\min \mathrm{Tr}(\varnothing^T L\varnothing)$$

$$\varnothing\in R^{n*k}$$

$$\mathrm{S.t}\, L=D-S$$

$$D=\mathrm{diag}\left(d_1,d_2,\dots,d_n\right) \tag{9}$$

$$\text{Ø}_{\text{ij}} = \begin{cases} \frac{1}{\sqrt{\text{vol}(C_j)}} & \text{if} \quad v_i \in C_j \\ 0 & \text{otherwise} \end{cases}$$

$$M = D^{-1} \tag{10}$$

In the case of this problem, the solution matrix ∅ of the specific vectors of k is the minimum nonzero particular value of the normalized Laplacian matrix $D^{-1}L$. In other words, k is the most significant specific value M covers, representing the solution in the hidden space. More importantly, the solution matrix Φ provides a perfect representation for obtaining the clustering.

Given the above, a higher modularity leads to a better partition structure; conversely, a lower normalized cut value enhances the two critical principles of graph classification, namely maximum integrity and minimum connection.

3.5 Graph embedding

Let $G=(V, E, X)$, where $V = \{v_i\} i = 1,2,..,n$ is formed of a set of graph nodes and $e_{\text{ij}} =< v_i, v_j >\in E$ represents a connection between the nodes. The topological structure of graph G is illustrated by adjacency matrix A, where $A_{\text{ij}} = 1$ if $\text{e}_{\text{ij}} \in E$ and otherwise $A_{\text{ij}} = 0$. $X \in R^{n*d}$ is the node attribute matrix, and d is the number of attributes. In addition, $x_i \in X$ shows the attributes of the content of each node v_i. The objective of the embedding problem is to map nodes $v_i \in V$ to low-dimensional vectors $\vec{z_i} \in R^d$, with a formal format $f : (A, X) \to Z$, where $z_i{}^T$ is the-i row of the $Z \in R^{n*d}$ matrix (n is the number of nodes, and d is the packing dimension). We assume that Z is the packing matrix, so the packings should preserve A's topology and content information, X.

4 Notations

Table 1, which includes various matrices, graph properties, and representation details relevant to the discussed methods, consolidates the essential symbols used in this paper.

5 The proposed method: DEGCN

Our proposed model aims to identify communities in attributed social networks using an efficient and interpretable embedding method by leveraging a parallel dual-graph convolutional neural network. This model consists of four main phases: the first phase is graph filtering; the second phase involves the calculation of modularity and Markov matrices; the third phase is network embedding using the parallel dual-graph convolutional network to obtain a new and meaningful representation; and the fourth phase is classification. Each phase's output seamlessly transitions to the next

Table 1 List of notations used in this paper

Descriptions	Symbols	Descriptions	Symbols
A similarity matrix	S	Graph adjacency matrix	A
The modularity value of (vi; vj)	Bij	Graph attribute matrix	X
The modularity evaluation metric	Q	Number of nodes in the graph	N
The pairwise node similarity value of (vi; vj)	Sij	Representations of nodes	Z
A degree matrix	D	Hidden dimensions	H
A Laplacian matrix	L	Reconstructed graph adjacency matrix	$\overline{A}$
A modularity matrix	B	Number of communities in the graph	K
A Markov matrix	M	Feature representation at layer $i+1$	$H^{[i+1]}$
Feature representation at layer i	H^i	The Activation function	$\sigma(0)$
Based on layer i	b^i	Weight at layer i	W^i

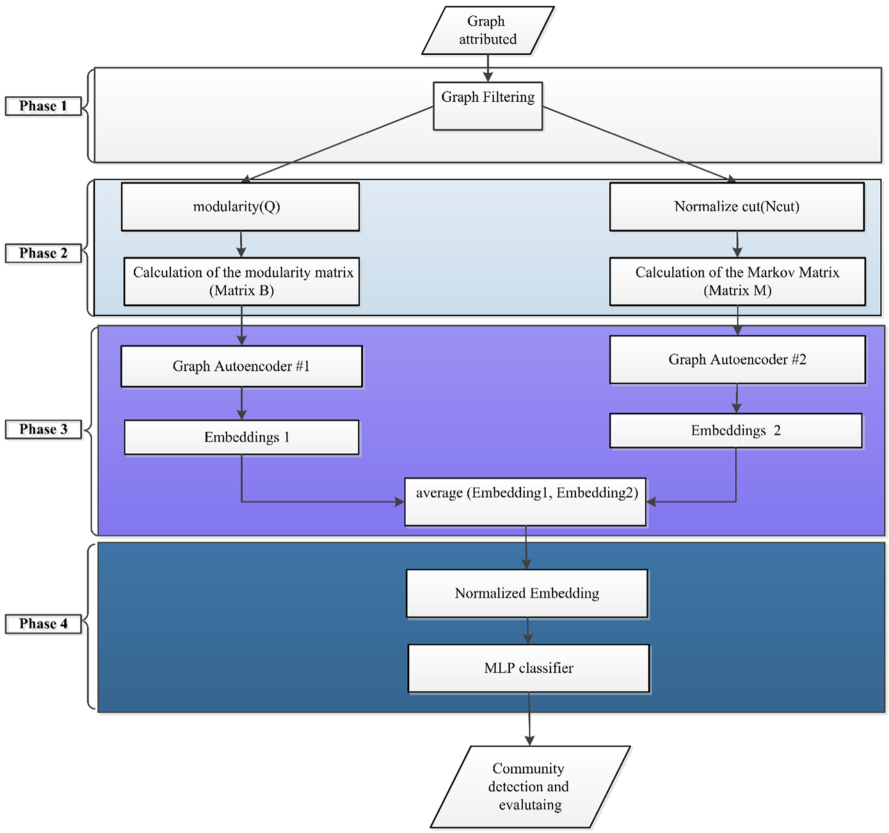

Fig. 1 Flowchart of the proposed method DEGCN

phase as input. Figure 1 depicts a detailed schematic of the proposed method. The following sections will delve deeper into the complexities of our approach.

5.1 Graph filtering

By applying the *k*-core algorithm, we filter the graph to remove nodes of lesser significance, commonly known as low-degree nodes. This process contributes to reducing the graph's size and complexity, thus optimizing the performance of subsequent algorithms applied to the graph for community detection. This technique emphasizes the more salient regions of the graph, streamlining computations. Specifically, a *k*-core is a maximal subset of the graph's nodes where each node is connected to at least *k* other nodes within the subgroup. For a node to be included in the *k*-core, its degree within this subset must be at least *k*. The *k*-core is computed; nodes with a degree less than *k* are removed. Subsequently, the degrees are recalculated, and the removal process is repeated. This iterative process continues until all nodes meet the *k*-core criterion. Each iteration has a computational complexity of O (E), where E represents the number of edges. Algorithm 1 provides the pseudocode for the *k*-core algorithm. Figure 2 illustrates a step-by-step visualization of a graph undergoing *k*-core decomposition, focusing on nodes with a degree less than 3 being iteratively removed. The following is an explanation for each iteration:

- All nodes are evaluated for degrees in the initial graph (iteration 0). Nodes 0 and 8 are identified as having degrees less than 3. These nodes are marked in red to indicate that they will be removed from the graph. This marking marks the beginning of the *k*-core decomposition process, which sets the stage for subsequent iterations by removing nodes that do not meet the minimum degree requirement.
- After removing nodes 0 and 8, the graph's structure changes, affecting the degrees of the adjacent nodes. In iteration 1, node 1 has a degree less than 3 (previously connected to node 0, impacting its degree). Node 1 is, therefore, marked in red and removed from the graph. This iteration further simplifies the graph, focusing the subsequent analysis on the remaining nodes.
- With node 1 removed, in iteration 2, the degree of node 5 (connected to node 1) is re-evaluated. The degree of node 5 decreases to less than three due to the

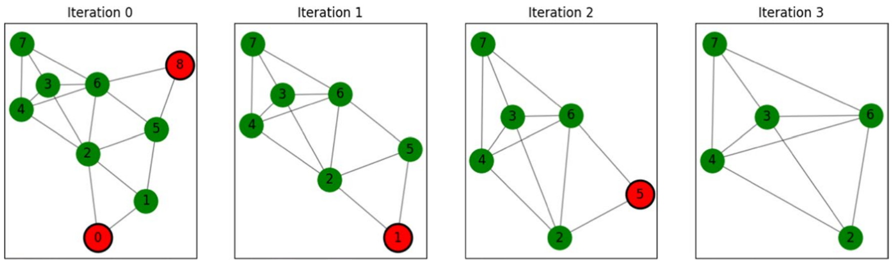

Fig. 2 A sample of the algorithm's 3-core output

removal of node 1, leading to its selection for removal. This iteration is marked in red. The graph's structure continues to be pruned to focus on the core nodes with higher connectivity. In this final iteration (iteration 3), no nodes are removed, indicating that all remaining nodes have a degree of 3 or more. Nodes 2, 3, 4, 6, and 7 form the stable core of the graph. This core represents the k-core of the graph, with each node satisfying the minimum degree requirement of 3. Such a conclusion is the step in this k-core decomposition process, highlighting the original network's central, most interconnected component.

Each iteration systematically reduces the graph by removing nodes with insufficient connectivity, eventually resulting in a simplified core that illustrates the most interconnected nodes. Given the explanations provided in this section, it is clear that the *k*-core algorithm defines a community based on density; employing this algorithm reduces the graph's size (thus expediting the community detection algorithm in subsequent steps) and enhances the community-centric orientation of graph neural networks. The dataset being analyzed influences this algorithm's choice of *k*. We determined $k=3$ this study was based on a trial-and-error approach.

Algorithm 1 *k*-core decomposition

```
Input: a graph G= ($V$, $E$), where $V$ is the set of vertices and $E$ is the set of links
Output: a graph $\acute{G} = (\acute{V}, \acute{E}), where\ \acute{V} \subseteq V\ and\ \acute{E} \subseteq E$
Begin
 For each $v_i \in V$ do
  If degree $v_i<k$
     Remove $v_i$ of V and $Relevant\ links$, $\acute{V} = V - v_i,\ \acute{E} = \acute{links}$
  Else
     Do not remove the $v_i$
  Update a set of $V$ and $E$ of Graph G
 End for
Return $\acute{V},\ \acute{E}$
End
```

5.2 Calculation of the modularity matrix and normalized cut matrix

This section calculates the modularity matrix (Matrix *B*) and the Markov matrix (Matrix *M*) of the filtered graph, which results from applying the 3-core algorithm. These calculations are based on Eqs. (2) and (10), as described in Sects. 3.4.1 and 3.4.2, respectively.

5.3 Network embedding

The goal of the learning phase is to attain a robust embedding of the data graph $G=(V, E, A, X)$. We use a graph autoencoder to process the entire graph and learn an effective embedding to achieve this. Figure 3 illustrates a portion of the workflow for the processing method. In the graph autoencoder, the encoder takes the graph's structure A and the node content X as inputs to learn the latent representation Z. Subsequently, the decoder reconstructs the graph's structure from Z. In a convolutional graph context, the graph autoencoder's objective is to embed each node in the graph into a lower-dimensional space. This process aims to aggregate the representations of similar nodes in this new feature space, focusing on developing a space-oriented approach to community-specific features. The graph autoencoder comprises two components, the encoder and the decoder, as elaborated below:

5.3.1 Encoder convolutional graph model

A graph convolutional network (GCN) [53] was developed as a graph encoder to represent graph A's structure and node X's content in an integrated framework. This GCN expands the convolution operation to the graph data in the spectral field and learns a stratified transformation through a spectral convolutional function $f\left(Z^{(l)}, A|w^{(l)}\right)$ in Eq. (11):

$$Z^{(l+1)} = f\left(Z^{(l)}, A|w^{(l)}\right) \tag{11}$$

$$Z^{(0)} = X \in R^{n*m}(n \text{ nodes and } m \text{ features})$$

where $Z^{(l)}$ is a convolution input, $Z^{(l+1)}$ is the convolution output, and $w^{(l)}$ is a matrix of filter parameters needed for learning in the neural network. If $f\left(Z^{(l)}, A|w^{(l)}\right)$ is well defined, we can construct an efficient deep convolutional neural network. Each layer of the GCN is expressed as $f\left(Z^{(l)}, A|w^{(l)}\right)$ using Eqs. (12) and (13):

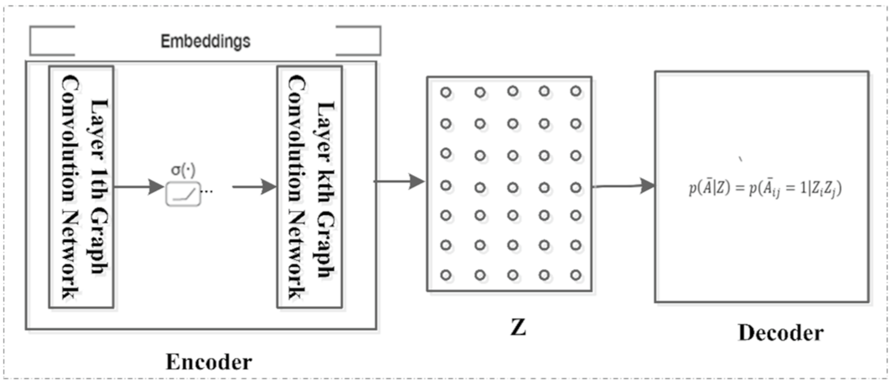
Embeddings
Layer 1th Graph Convolution Network
σ(·)
Layer kth Graph Convolution Network
$p(\bar{A}|Z) = p(\bar{A}_{ij} = 1|Z_i Z_j)$
Z
Decoder
Encoder

Fig. 3 Workflow scheme of the proposed graph autoencoder

$$f\left(Z^{(l)}, A|w^{(l)}\right) = \emptyset\left(\tilde{D}^{-\frac{1}{2}}\widetilde{A}\tilde{D}^{-\frac{1}{2}}Z^{(l)}w^{(l)}\right) \tag{12}$$

$$\tilde{A} = A + I, \tilde{D}_{ii} = \sum_{j} A_{ij} \tag{13}$$

where I is an identity matrix of A and $\emptyset$ is a nonlinear activation function such as $\text{Relu}(t) = \max(0, t)$or $\text{sigmoid}(t) = \frac{1}{1+e^t}$. In this paper, the encoder graph $G(X, A)$ utilizes a two-layer GCN given the dataset type. The GCN organizes the hidden features of the graph at several layers to meet the requirements of community detection. Equations (14) and (15) present the encoder graph equations for the two-layer GCN:

$$Z^{(1)} = f_{\text{ReLU}}\left(X, A|W^{(0)}\right) \tag{14}$$

$$Z^{(2)} = f_{\text{ReLU}}\left(Z^{(1)}, A|W^{(1)}\right) \tag{15}$$

The Relu (0) function refers to activation functions applied to the first and second layers. $W^{(0)}$ and $W^{(1)}$ are the weights associated with each layer. The activation function, Relu (0), is pivotal in graph representation learning. The encoder convolution graph, denoted as $G(X, A) = q\ (Z|X, A)$, translates the graph structure and node content into the representation $Z^{(2)}$.

5.3.2 Decoder model

The proposed decoding model is used to reconstruct graph data. We can rebuild a graph structure, content information X, or both. Here, reconstruction of the graph structure is recommended, which gives us a higher level of flexibility so our algorithm preserves its functionality even if content information X is unavailable. A decoder $p(\overline{A}|Z)$ predicts whether there is a connection between the two nodes of a connection. As per Eqs. (16) and (17), we trained a connection prediction layer based on graph embedding:

$$p\left(\overline{A}|Z\right) = \prod_{i=1}^{n}\prod_{j=1}^{n} p(\overline{A}_{\text{ij}}|z_{i,}z_j) \tag{16}$$

$$p\left(\overline{A}_{\text{ij}} = 1|z_{i,}z_j\right) = \text{sigmod}\left(z_i^T, z_j\right) \tag{17}$$

The embedding of Z and $\overline{A}$ The reconstructed graphs are given in Eq. (18):

$$\overline{A} = \text{sigmod}\left(ZZ^T\right), \text{here } Z = q(Z|X, A) \tag{18}$$

Based on the descriptions provided in Sects. 4.3.1 and 4.3.2 of graph convolution networks for the propagation of feature representations to subsequent layers, Eq. (19) is used, and nonlinear activation functions are employed to represent nonlinear features. Consequently, the feature representations at layer zero are essentially the input features (X), as described by Eq. (20):

$$H^{[i+1]} = \sigma\left(W^{i}H^{i} + b^{i}\right) \tag{19}$$

where $H^{[i+1]}$ represents the feature representation at layer $i+1$, emerging as the output of the activation function σ. This function is crucial for introducing non-linearity to the networks processing, allowing more complex patterns to be captured. Concurrently, W^{i} serves as the weight matrix at layer i, crucially influencing how the features from the previous layer affect the current layer's outputs. Furthermore, b^{i} the bias at the same layer i is added to each neuron, enhancing the model's flexibility and ability to learn diverse patterns. This combination of weights and biases, adjusted through training, determines the effectiveness of the network in feature learning and transformation across successive layers.

$$H^{[1]} = \sigma\left(W_{[0]}^{T}X + b^{[0]}\right) \tag{20}$$

where $W_{[0]}^{T}$ The weight matrix's transpose is associated with the first layer, and X denotes the input feature matrix. The term $b^{[0]}$ is the bias vector for the first layer. The function σ signifies the activation function, which is applied element-wise. This equation effectively captures how the weights and biases linearly transform the input data X before being passed through the activation function to produce the first layer output, $H^{[1]}$, which then serves as input to subsequent layers in the network.

This section uses a spectral graph convolutional network to disseminate information through the eigen decomposition of the graph Laplacian matrix. Consequently, Eq. (19) is implemented using this network according to Eq. (21). This equation integrates the adjacency matrix and the input features of the nodes into the forward propagation equation. Therefore, this technique enables the model to understand the nodes' features through the eigen decomposition interconnections. Furthermore, the adjacency matrix is added to this propagation process:

$$H^{[i+1]} = \sigma\left(W^{[i]}H^{[i]}A^{*}\right) \tag{21}$$

In this paper, matrix $A*$ is a renormalization of A (the adjacency matrix) designed to counteract the vanishing gradient issue. Instead of using a matrix $H^{[i]}$ to create a community-oriented representation with reduced dimensions, we utilize two specific matrices: the modularity matrix (B) and the Markov matrix (M) as input for graph autoencoder number 1 and graph autoencoder number 2, respectively, as shown in Fig. 4. Figure 4 illustrates the architecture of our proposed community detection model using dual embedding-based graph convolution networks (DEGCNs), which integrate both the graph structure and node features.

This dual-graph autoencoder proposed setup facilitates parallel processing and allows for comparing different matrix influences on the interpretation of graph structure. The first graph autoencoder's (#1) decoder, composed of a two-layer graph convolutional neural network, processes these inputs according to Eq. (22), producing a representation noted as Z_1. Simultaneously, the decoder of the second

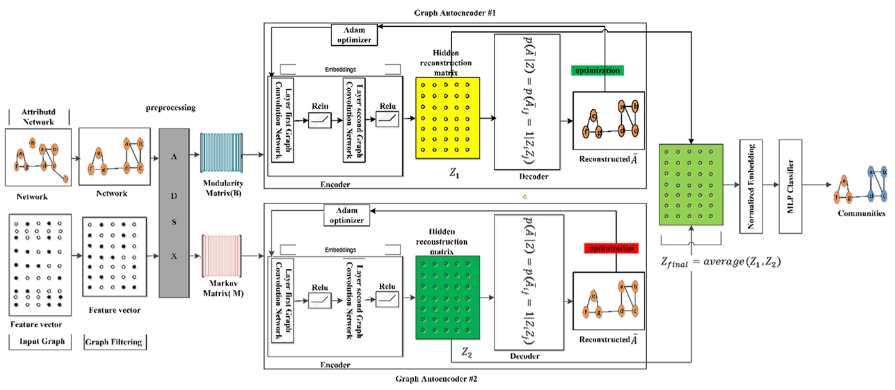

Fig. 4 DEGCN framework for community detection in attributed social networks

graph autoencoder (#2), also structured as a two-layer GCN, processes inputs under Eq. (23) to generate another representation, denoted as Z_2. These representations are then combined according to Eq. (24):

$$\text{embeding1} = Z_1 = f(M, A) = \text{Relu}\left(\tilde{A} \cdot \text{Relu}\left(\tilde{A} \cdot M \cdot W^0\right) W^1\right) \tag{22}$$

This embedding is computed using matrices M (Markov matrix), A (adjacency matrix), and weight matrices W^0 and W^1. Here, $\widetilde{A}$ is the normalization of the adjacency matrix A, applied to modulate the influence of neighboring nodes differently at two stages of the transformation.

$$\text{embeding2} = Z_2 = f(B, A) = \text{Relu}\left(\tilde{A} \cdot \text{Relu}\left(\tilde{A} \cdot B \cdot W^0\right) W^1\right) \tag{23}$$

This embedding is similar to Z_1 but uses a modularity matrix, B instead of M:

$$\text{embeding result} = Z_{\text{final}} = \text{average}(Z_1, Z_2) \tag{24}$$

This operation combines the information from both embeddings, potentially capturing a more comprehensive representation by integrating matrix M and B with different perspectives or feature interpretations.

5.3.2.1 Loss function and optimization The Adam optimizer, a well-known optimizer for deep learning, trains the overall end-to-end deep neural network. The loss function is the cross-entropy loss between the predicted community labels and the partial ground-truth labels. The Adam optimizer in the diagram optimizes the models' weights by minimizing the loss during training. In the context of a graph autoencoder, as depicted in Fig. 4, the Adam optimizer helps adjust the neural network weights to effectively reconstruct the graph's adjacency matrix from the learned embeddings. The Adam optimizer minimizes the difference (loss) between the original and reconstructed adjacency matrix from the embedded. The model can produce more accurate embeddings of the original graph data by reducing this loss. The graph data reconstruction error for a self-encoder graph is minimized using Eq. (25):

$$\mathcal{L}_0 = E_{q(Z|(x,A))}[\log_p(\overline{A}|Z)] \quad (25)$$

This loss function evaluates how effectively the model can reconstruct the adjacency matrix based on the latent representations (Z) extracted from the data (X) and the original adjacency matrix (A). This setup is typical when estimating how well the model reconstructs the adjacency matrix based on the latent representations.

The function q ($Z|X$, A) indicates a probabilistic encoding of the graph data and adjacency matrix into a latent space, and p $(\overline{A}|Z)$ represents the probabilistic decoding from the latent space to the adjacency matrix. Using the log probability in the loss function is a common choice in machine learning for stability and calculation efficiency, particularly under cross-entropy loss.

5.4 Node classification

In this phase, we apply min–max scaling to normalize the Z_{final} feature vectors obtained in the previous phase. This normalization process scales the data values to zero and one, ensuring uniformity across all features. Such uniformity is crucial as it enhances the speed of convergence and the stability of the MLP classifier, thereby improving the overall learning efficiency. The MLP classifier is designed with three layers, containing 128, 64, and 32 neurons, respectively. The model is set to run for 500 iterations as specified by the maximum iteration parameter. Relu is chosen as the activation function for the hidden layers due to its effectiveness and simplicity, making it a popular option in deep learning models.

5.5 Time complexity analysis

Table 2 provides a general pseudocode of the program, detailing each line's time complexity and accompanying explanations. Loading the graph $G=(V, E, A, X)$ has a complexity of O $(N+E)$, and applying k-core filtering is O (E). The GCN model is initialized with O (1). For each epoch, training the GCN model takes O $(E+N\times D)$, extracting embeddings is O $(N\times D)$, splitting data is $O(N)$, performing classification with MLP is O (N_{train}), and making predictions is O (N_{test}). The computational evaluation metric is O (N_{test}). Thus, the training phase significantly contributes to the overall complexity, yielding a final O (Epoch$\times(E+N\times D)$) complexity. Table 3 compares community detection schemes using GCNs—characteristics, properties, and computational complexity. The computational complexity of an algorithm is a critical factor in determining its efficiency, especially when dealing with large datasets. Comparatively, several methods in Table 2, such as LGNN, MRFasGCN, and CayleyNet, have a complexity of O (m). While this seems linear and comparable to the DEGCN, the actual efficiency would depend on what the variable m represents in their respective contexts. If m is significantly larger than the sum of the edges and nodes multiplied by the number of dimensions, the DEGCN will have a computational advantage.

Table 2 Pseudocode of the proposed method

```
Requires: Graph G= (V, E, X), D: the dimension of the latent variable, N: is the number of nodes, E: is the number of edges, X: is features vector per node, N_train: is the size of the training set, N_test: the size of the test set, Epoch: Number of training cycles, D: Features per node, Ensure: $Z \in R^{N*D}$
1. Begin:
  2. For each dataset
    3. Begin:
      4. Load Graph G
      5. Apply k-core filtering to the dataset using Algorithm 1
      6. Initialize the GCN model and other related setup
      7. Train model
      8. For each Epoch:
        9. Begin:
          10. Extract embeddings using a trained GCN model by Eq (23), Eq(24), Eq(25)
          11. Split data into Train set, Test set, and Validation set
          12. Perform Classification with MLP algorithm on Train embeddings
          13. Make Predictions on Test embeddings
          14. Compute evaluation metrics using Eq(26),Eq(27), Eq(28), Eq(29) , Eq(30)
        15. End
      16. End
      17.return: $Z \in R^{N*D}$
18. End
```

6 Experiment

In this section, we have conducted several meticulous experiments based on real-world scenarios and valid datasets to make a fair comparison of DEGCN with state-of-the-art related works. Here, we describe the assessment metric, baseline methodology, experimental setting, and ablation studies.

6.1 Experimental settings

6.1.1 Datasets

We sourced datasets from practical applications for our community detection experiments, allowing a comprehensive assessment of our proposed methods. Table 4 [57] provides the statistical details of the three datasets used in this study. These datasets are citation networks in which nodes represent papers, edges represent citation relationships, attributes are word packet representations of paper abstracts, and labels correspond to paper topics.

Table 3 Summary of GCN-based community detection methods

Method	Learning type	Convolution	Clustering algorithm	Other characters	Complexity	Input
LGNN [35]	Supervised	First-order line graph	–	Edge feature	$O\ (m)$	A, X
MRFasGCN [27]	Semi-supervised	First-order + mean field approximate	–	GCN + eMRF	$O\ (m)$	A, X
CayleyNet [39]		Laplacian smoothing filter	–	Cayley polynomial	$O\ (m)$	
CommDGI [11]	Unsupervised	First-order + sampling	–	Join optimization	–	A, X
NOCD [54]		First-order	–	GCN + BP	–	
GCLN [37]		First-order	K-means	Unet architecture	–	
IPGDN [38]		First order + disentangled representation	K-means	HSIC as regularizer	$O\ (m)$	
AGC [55]		k-order + Laplacian smoothing filter	Spectral clustering	–	$O\ (\mathrm{n}^2\mathrm{dt}+\mathrm{mdt})$	
AGE [56]		Laplacian smoothing filter	Spectral clustering	Adaptive learning	–	
DEGCN	Supervised	K-order	MLP	GCN	$O\ (E+N\times D)$	$\boldsymbol{A, X}$

Table 4 Summary of real-world benchmarks on datasets

Dataset	#Nodes	#Edges	#Node attributes	Num. of communities
Cora [58]	2708	5429	1433	7
CiteSeer [58]	3312	4715	3703	6
PubMed [59]	19,717	44,338	500	3

6.1.2 Evaluation metrics

This section introduces a set of qualitative measures used for assessing community detection methods, categorized into performance and goodness measures. Performance measures examine the quality of the community obtained via the algorithm compared with actual communities. In addition, goodness measures assess the structural specifications of the detected communities [60]. We used six measures of normalized mutual information, the rand-adjusted index, accuracy, precision, and F1 score, to assess the proposed method. Higher scores on all the measures represent better results. In the following sections, we further discuss these measures:

- Normalized mutual information

The normalized mutual information computed through Eq. (26) represents the similarity of the final community set obtained by the proposed algorithm to the correct community [60].

$$\mathrm{NMI} = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} n_{ij}\ln(n_{ij}\cdot n/n_i^c\cdot n_j^c)}{\sqrt{\left(\sum_{i=1}^{k} n_i^c \ln\left(\frac{n_i^c}{n}\right)\right)\left(\sum_{j=1}^{k} n_j^{\acute{C}} \ln\left(n_j^{\acute{C}}/n\right)\right)}} \tag{26}$$

where k is the number of communities, n is the number of nodes, n_{ij} is the number of nodes in the optimized community set i such that the proposed community set is in community j, n_i^c, the number of nodes in the community i, which is in the optimized community set, and n_j^c is the number of nodes in community j.

- Accuracy

It determines the authenticity of the community setting, and similar to NMI, computing this measure requires an optimum community setting Eq. (27) [60].

$$\mathrm{ACC} = \frac{\sum_{i=1}^{n} k\left(C_i, \mathrm{PM}\left(\acute{C}_i\right)\right)}{n} \tag{27}$$

where n is the number of groups, and for a specific group, i and $\acute{C}_i$, $\underset{i}{C}$ are the communities of node i in optimum and recommended community settings. $K(x, y)$ is a function equal to 1 when $x=y$ and 0 otherwise.

- Rand index adjusted

The rand index adjusted (RI) [61] is a measure that compares the results of partitioning by a method to actual partitions. In addition, the RI was used to compare the results of the two clustering methods, as shown in Eq. (28):

$$\mathrm{RI}(Y, C) = \frac{(a+b)}{\binom{n}{2}} \tag{28}$$

where n is the total number of nodes, Y and C represent two different clusters, the number of pairs in a similar cluster in C and Y, respectively, and b is the number of pairs in dissimilar groups. Notably, the RI is in the [0, 1] interval, equal to 1 when two sets of clusters are identical and equal to 0 when the two sets of clusters are entirely different.

- Precision

It calculates the accuracy of the community and the percentage of nodes in the detected community that belong to the actual community and is formulated according to Eq. (29):

$$\mathrm{Precision}\left(C_i, C_i^*\right) = \frac{C_i \cap C_i^*}{\left|C_i^*\right|} \tag{29}$$

where C_i^* is the detected community and C_i is the actual community.

- F1-scores

The F1-score of community detection is the harmonic mean of recall and precision, which is formulated as Eq. (30):

$$F1 - \mathrm{scores}\left(C_i, C_i^*\right) = 2 \cdot \frac{\mathrm{precision}\left(C_i, C_i^*\right) \cdot \mathrm{recall}\left(C_i, C_i^*\right)}{\mathrm{precision}\left(C_i, C_i^*\right) + \mathrm{recall}\left(C_i, C_i^*\right)} \tag{30}$$

where C_i^* is the detected community and C_i is the actual community.

6.1.3 Parameter settings

We selected 20 nodes from each class to form the training set, 500 for the validation set, and 1000 for the test set. We conducted all experiments using a two-layer GCN configuration. The first layer comprises 64 neurons, with each subsequent layer in the contracting path halving the number of neurons from the previous layer. We employed the widely used Adam optimizer for training and conducted experiments using Tensor Flow and PyTorch. We set the learning rate at 0.01 and dynamically optimized it using a scheduler that reduced the rate when the loss plateaued, thereby promoting more stable convergence.

Additionally, we set the dropout rate at 0.5 and the maximum number of epochs at 200. We applied the Relu activation function after every graph convolutional operation. We terminated the training if the loss function decreased over 10 consecutive epochs. We randomly selected the initial weights of the two GCN layers in the DEGCN from a uniform distribution. We repeated each experiment ten times and reported the average scores below. Table 5 summarizes the parameter settings used in our experiments, detailing the names of the parameters and their corresponding values. We implemented the research on a system with an AMD Ryzen 71700X Eight-Core Processor at 3.77 GHz and 32 gigabytes of RAM, using Python version 3.7 and Anaconda3-2019.03-Linux-x86_64.

6.1.4 Experimental results and analysis

This subsection describes the experimental results analyzed from multiple evaluation perspectives. To validate the efficiency of our proposed model, we examined the Cora, CiteSeer, and PubMed medium–scale datasets. We compared our proposed technique with three baseline categories of established methods to gain a comprehensive understanding. The following sections detail these comparative methods, which serve as our reference points:

- *Node Feature-Based Methods*: This category predominantly relies on individual nodes' unique attributes or characteristics. Methods such as k-means and spectral clustering, known as spectral f, exemplify this approach. These techniques construct a similarity matrix from node features, typically using a linear kernel.
  - *Graph Structure-Based Methods*: This category focuses on the graph's intrinsic structure. Techniques such as spectral clustering (Spectral-g) use the node adjacency matrix to construct the similarity matrix. DeepWalk [14] is another important method in this group; it is excellent at learning graph embeddings. DNGR [62] combines the benefits of spectral graph clustering with deep neural networks to understand complex graph representations. vGraph[63] is a probabilistic generative model to learn community membership and node representation collaboratively. Graph Encoder [64] learns graph embedding for spectral graph clustering.
- *Hybrid Methods*: These methods combine node attributes and graph structure to improve community detection. Although this approach generally increases computational complexity, it often results in better outcomes than those that use only node or structure information. Within this domain, several graph autoencoder variants have emerged:
  - GAE [65]: A model that utilizes neural networks to learn graph representations.
  - VGAE [65]: An advancement of GAEs that applies a variational inference framework.

Table 5 Detailed parameter setting

Datasets	Training Epoch	Learning rate	Activation function	Weight decay	Optimizer	GCN layers	Dropout rate	#Train/validation/test node
Cora	200	0.01	Relu	5e-3	Adam	64/32	0.5	140/500/1000
CiteSeer	200	0.01	Relu	5e-3	Adam	64/32	0.5	120/500/1000
PubMed	200	0.01	Relu	5e-3	Adam	64/32	0.5	60/500/1000

- MGAE [18]: This technique improves representation by marginalizing specific graph properties.
- ARGA [66]: This method uses adversarial training to improve the regularization of graph embeddings.
- ARVGA [66]: This technique incorporates vibrational regularization into encoding.
- DAEGC [67]: This approach uses deep autoencoders to reconstruct the graph's adjacency matrix.
- AGE [56]: A model that uses a two-stage process to enhance graph-based learning tasks.
- AGC [55]: A model that leverages high-order graph convolution to understand a graph's global structure effectively
- DBGAN [68] and GALA [69] are two new ways to use graph neural networks to do two very different things: group nodes together and embed node features.
- CommDGI [11] and GC-VGE [70]: These graph neural network models optimize the simultaneous learning of node embeddings and cluster assignments.
- TADW [71]: A model that employs matrix factorization for network representation learning, representing a distinct approach to addressing this issue.
- RMSC [72]: A robust multi-view spectral clustering method via low rank and sparse decomposition.
- RTM [72]: A model that Learns how each document's topic is distributed from text and citation.
- GMIM [73]: A model that utilizes a mutual information maximization approach for node embedding.
- DGVAE [74]: A model that presents a graph variational generative model that uses the Dirichlet distributions as priors on the latent variables.
- BernNet [75]GCN: This technique uses a graph convolutional neural network framework based on the Bernstein polynomial approximation of order K.
- WC-GCN [76]: This technique utilizes a graph convolutional neural network framework.
- LGNN [35]: This method is a neural network model explicitly designed for graph data.
- MRFasGCN [27]: is a model that combines a GCN with the Markov random field statistical model for community detection.

Tables 6, 7 and 8 comprehensively compare the proposed method with baseline community detection methods based on their performance metrics. These metrics include accuracy (ACC %), normalized mutual information (NMI %), adjusted Rand index (ARI %), F1-score (F1%), and precision (P %). The compared approaches are often categorized into three groups based on the type of learning: supervised, semi-supervised, and unsupervised. Furthermore, these strategies are classified into three groups based on the input type: features, graph topology, or a hybrid of both.

Table 6 presents the proposed method with the highest F1 and precision scores, 83.19% and 83.70%, respectively, and an impressive NMI of 69.51%, demonstrating its high capability in identifying community structures. Methods such as GMIM,

Table 6 Performance comparison of different community detection methods on the Cora dataset

Name of methods	Learning type	Input	ACC%	NMI%	ARI%	F1%	P%
K-means	Unsupervised	Feature	49.2	32.1	23.0	36.8	36.9
Spectral-F [77]			34.7	14.7	7.1	–	–
Spectral-G [77]		Graph	31.46	9.69	0.35	29.67	18.07
DeepWalk [14]			56.20	39.87	32.18	47.6	5.48
Graph encoder [78]			32.5	10.9	0.6	29.8	18.2
DNGR [62]			44.39	33.31	15.86	34.68	27.86
vGraph [63]			28.7	34.5	31.2	30.5	–
TADW [71]		Feature and graph	55.00	36.59	26.40	41.52	36.50
GAE [65]			60.34	44.85	36.73	58.72	61.39
VGAE [65]			63.56	47.45	39.42	63.75	65.64
MGAE [18]			63.43	45.57	38.01	38.01	–
ARGE [66]			60.84	42.21	36.88	60.49	63.38
ARVGA [66]			62.83	45.93	38.00	63.17	64.80
DGVAE [74]			64.42	47.64	38.42	62.69	64.90
AGC [55]			68.92	53.68	48.6	65.61	–
CommDGI [11]			69.8	57.9	50.2	68.4	–
DAEGC [67]			70.4	52.8	49.6	68.2	–
GC-VGE [70]			70.67	53.57	48.15	69.48	70.51
GALA [69]			72.42	53.96	47.22	–	–
DBGAN [68]			74.6	57.7	53.2	–	–
GMIM [73]			74.8	56.0	54.0	–	–
AGE [56]			76.8	60.7	56.5	–	–
MRFasGCN [27]	Semi-supervised		84.3	66.2	–	–	–
BernNet GCN [75]	Supervised		41.06	68.78	–	–	–
LGNN [35]			79.04	–	–	79.04	–
WC-GCN [76]			79.39	–	–	75.32	–
DEGCN(proposed method)	Supervised		**83.03**	**69.51**	**66.09**	**83.19**	**83.70**

The best results are in bold; '–' indicates that runtime exceeds 24 h or out of memory

DBGAN, and AGE also demonstrate strong performance across multiple metrics, reflecting advancements in graph neural networks and deep learning techniques for community detection.

In contrast, traditional methods like *K*-means and spectral clustering exhibit lower performance, highlighting the superiority of modern techniques. For instance, K-means only achieves an ACC of 49.2% and an ARI of 23.0%, while newer methods like LGNN reach ACC values above 79%. Supervised methods such as LGNN, BernNet GCN, WC-GCN, and the semi-supervised MRFasGCN demonstrate remarkable performance on the Cora dataset. LGNN achieves a high ACC of 79.04% and an ARI of 79.04%, indicating its efficiency in leveraging labeled data for community detection. BernNet GCN, which incorporates a Bernoulli model,

Table 7 Performance comparison of different community detection methods on the PubMed dataset

Name of methods	Learning type	Input	ACC%	NMI%	ARI%	F1%	P%
K-means	Unsupervised	Feature	55.59	24.34	21.54	56.04	46.08
Spectral-F [77]			60.20	30.90	27.7	–	–
Spectral-G [77]		Graph	37.98	10.30	26.67	50.54	0.02
DeepWalk [14]			64.98	26.44	27.42	63.46	65.24
Graph encoder [11]			53.1	20.9	18.4	50.6	45.6
DNGR [62]			25.53	20.11	8.29	15.57	19.26
vGraph [79]			26.00	22.40	18.50	33.20	–
TADW [71]		Feature and graph	46.82	9.47	5.78	51.22	38.34
GAE [65]			64.43	24.85	23.57	64.07	65.26
VGAE [65]			64.67	23.94	23.41	64.77	64.53
MGAE [18]			43.88	8.16	3.98	41.98	–
ARGA [66]			65.07	29.23	26.79	64.11	69.27
ARVGA [66]			62.01	26.62	22.46	61.66	68.41
DGVAE [74]			67.56	28.72	24.92	64.35	67.10
AGC [55]			69.78	31.59	31.19	68.72	–
CommDGI [11]			69.90	35.70	29.2	69.2	–
DAEGC [67]			67.10	26.60	27.8	65.9	–
GC-VGE [70]			68.18	29.70	29.76	66.87	69.39
GALA [69]			69.39	32.73	32.1	–	–
DBGAN [68]			69.40	32.40	32.7	–	–
GMIM [73]			70.87	32.43	33.25	69.19	70.83
AGE [56]			71.1	31.6	33.4	–	–
MRFasGCN [27]	Semi-supervised		79.6	40.7	–	–	–
BernNet GCN [75]	Supervised		61.25	51.40	–	–	–
LGNN [35]			72.64	–	–	72.64	–
WC–GC [76]			79.41	–	–	73.75	–
DEGCN(proposed method)	**Supervised**		**81.34**	**53.71**	**51.07**	**81.30**	**81.33**

The best results are in bold; '–' indicates that runtime exceeds 24 h or out of memory

excels in NMI with a value of 68.78%, demonstrating its strength in mutual information metrics. WC-GCN also performs well, with an ACC of 79.39% and an ARI of 75.32%, highlighting its effectiveness in graph-based clustering. The semi-supervised MRFasGCN stands out with the highest ACC of 84.3% and a robust NMI of 66.2%, demonstrating the power of integrating Markov Random Fields with GCNs. Notably, the proposed DEGCN method achieves the highest ACC among the supervised methods, showing its exceptional accuracy even among supervised techniques. The proposed method outperforms all in F1 and precision, scoring 83.19% and 83.70%, respectively. It also achieves a high NMI of 69.51%, making it the most robust method overall for community detection in this dataset.

Table 8 Performance comparison of different community detection methods on the CiteSeer dataset

Name of methods	Learning type	Input	ACC%	NMI%	ARI%	F1%	P%
K-means	Unsupervised	Feature	54.0	30.5	40.9	40.5	27.9
Spectral-F [77]			23.9	5.6	29.9	17.9	0.010
DeepWalk [14]		Graph	32.7	8.8	27.0	24.8	9.2
Graph encoder[11]			22.5	3.3	30.1	17.9	0.010
DNGR [62]			32.6	18.0	30.0	20.0	4.4
RTM [72]			45.1	23.9	34.2	34.9	20.3
RMSC [72]			29.5	13.9	32.0	20.4	4.9
TADW [71]		Feature and graph	45.5	29.1	41.4	31.2	22.8
GAE [65]			40.8	17.6	37.2	41.8	12.4
VGAE [65]			34.4	15.6	30.8	34.9	9.3
MGAE [18]			43.88	8.16	39.8	41.98	–
ARGA [66]			57.3	35.0	45.6	57.3	34.1
ARVGA [66]			54.4	26.1	52.9	54.9	24.5
AGE [56]			70.2	44.8	45.7	–	–
MRFasGCN [27]	Semi-supervised		73.2	46.3	–	–	–
BernNet GCN [75]	Supervised		72.32	58.01	–	–	–
LGNN [35]			73.15	–	–	73.15	–
			73.2	46.3	–	–	–
WC-GCN [76]			75.18	–	–	69.33	–
DEGCN (proposed method)	**Supervised**		**75.45**	**59.09**	**53.99**	**74.43**	**75.10**

The best results are in bold, '–' indicating that runtime exceeds 24 h or is out of memory

To make a fair comparison with other related works, we repeated the experiments on two different datasets, the PubMed dataset and the CiteSeer dataset. We present the results and figures of this new evaluation in Tables 7 and 8, respectively.

Table 7 shows that the proposed DEGCN method performs better than the other methods baseline, especially compared to supervised methods. DEGCN achieves the highest accuracy (ACC) of 81.34%, significantly surpassing MRFasGCN at 79.6% and WC-GC at 79.41%. Regarding normalized mutual information (NMI), DEGCN scores 53.71%, notably higher than BernNet GCN at 51.40%, and LGNN, which does not report NMI but has a high accuracy. DEGCN also leads in the adjusted Rand index (ARI) with 51.07%, compared to GMIM's 33.25% and AGE's 33.4%, indicating a closer match to the actual community structures. Furthermore, DEGCN's F1-score of 81.30% outshines WC-GC's 73.75%, reflecting a better balance between precision and recall. DEGCN achieves 81.33% in precision (%), significantly higher than ARGA's 69.27% and GMIM's 70.83%, indicating fewer false positives. These results show that DEGCN is solid and reliable, especially compared to supervised methods. They also show that it locates community structures in the PubMed dataset better. The consistent outperformance across all key metrics highlights DEGCN's effective integration of feature and graph information, setting it apart as the most effective method for community detection in this context.

Based on the analysis of the results presented in Table 8, the proposed method achieves the highest accuracy (75.45%), marginally outperforming WC-GCN (75.18%) and significantly surpassing other supervised methods like MRFasGCN (73.2%), LGNN (73.15%), and BernNet GCN (72.32%). Regarding normalized mutual information (NMI), DEGCN leads with 59.09%, closely followed by Bern-Net GCN at 58.01%, indicating better mutual information capture between predicted and actual community structures. The adjusted Rand index (ARI) for DEGCN is 53.99%, surpassing ARVGA (52.9%) and AGE (45.7%), showing a higher agreement with the actual partitioning of the data. Furthermore, DEGCN's F1-score of 74.43% outshines LGNN's 73.15% and WC-GCN's 69.33%, reflecting a superior balance between precision and recall. In precision, DEGCN achieves 75.10%, the highest among all methods, including WC-GCN (69.33%) and ARGA (34.1%), indicating a higher rate of correctly identified positives. These results underscore DEGCN's robustness and reliability, particularly among supervised methods, making it the most effective for accurately detecting community structures in the CiteSeer dataset. The consistent outperformance across key metrics highlights DEGCN's superior feature and graph information integration, setting it apart as the best-performing method in this analysis.

6.2 Ablation studies

To demonstrate the efficacy of our proposed method, we conducted ablation studies on the DEGCN model. An EGCN model contains more than just the topology and feature modules. The BEGCN is a model that contains only the topology module. The MEGCN model contains a feature module. The DEGCN-K model does not include a K-Kore algorithm for preprocessing. We evaluated accuracy (ACC) and normalized mutual information (NMI). Table 9 summarizes the percentage results from the ablation experiments below, with the best results highlighted in bold. Figures 5, 6, and 7 demonstrate the performances of various models on the CiteSeer, PubMed, and Cora datasets, respectively. They compared models based on two key metrics: ACC% and NMI%. Each figure visually represents how different graph neural network models perform, highlighting variations in model effectiveness in accuracy and the ability

Table 9 Percentage of ablation experimental results (best result bold)

Dataset	Metric (%)	EGCN	BEGCN	MEGCN	DEGCN-K	DEGCN
Cora	ACC	64.65	69.25	68.5	80.47	**83.03**
	NMI	45.23	51.26	50.36	66.35	**69.51**
PubMed	ACC	62.87	68.13	74.15	78.12	**81.34**
	NMI	27.15	34.14	38.21	50.28	**53.71**
CiteSeer	ACC	51.62	62.06	61.63	72.18	**75.45**
	NMI	24.22	31.45	31.28	56.25	**59.09**

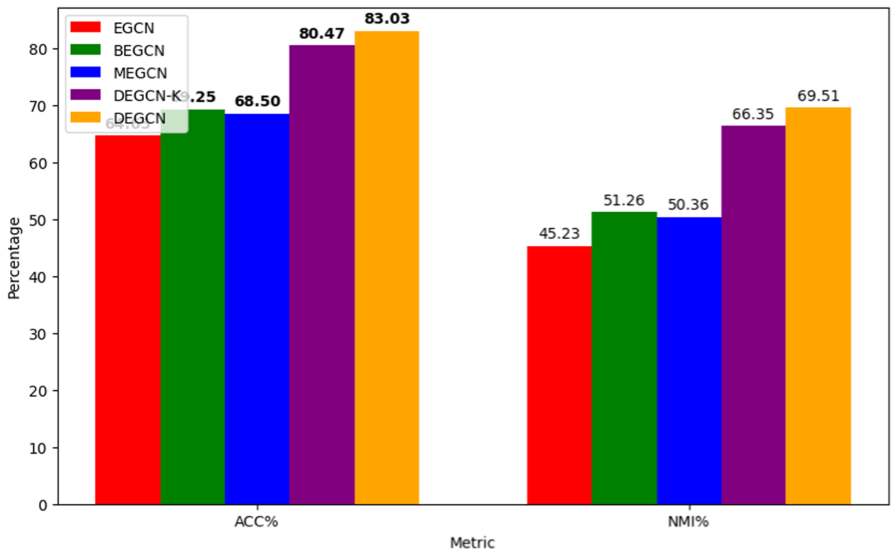

Fig. 5 Performance metrics comparison for the Cora dataset

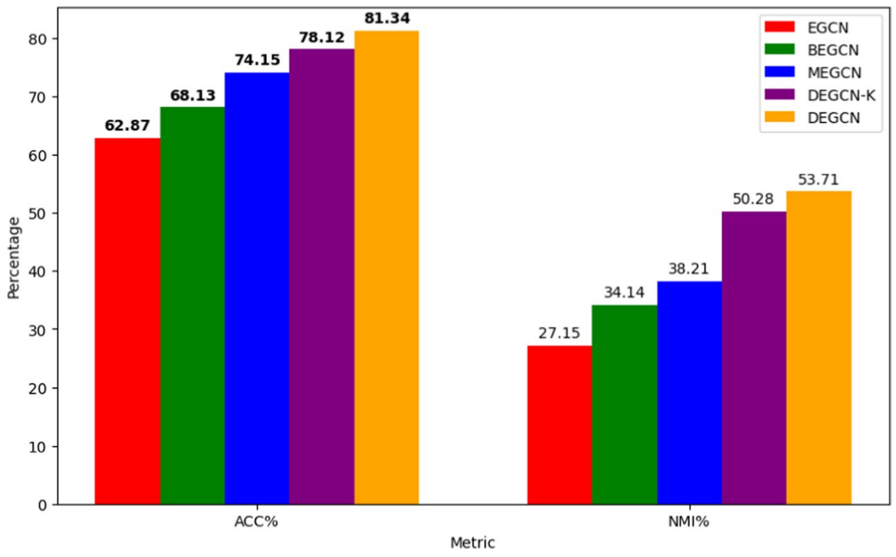

Fig. 6 Comparison of performance metrics for the PubMed dataset

to capture mutual information across these diverse datasets. This analysis will focus on the differences in ACC and normalized NMI performance metrics among various model configurations.

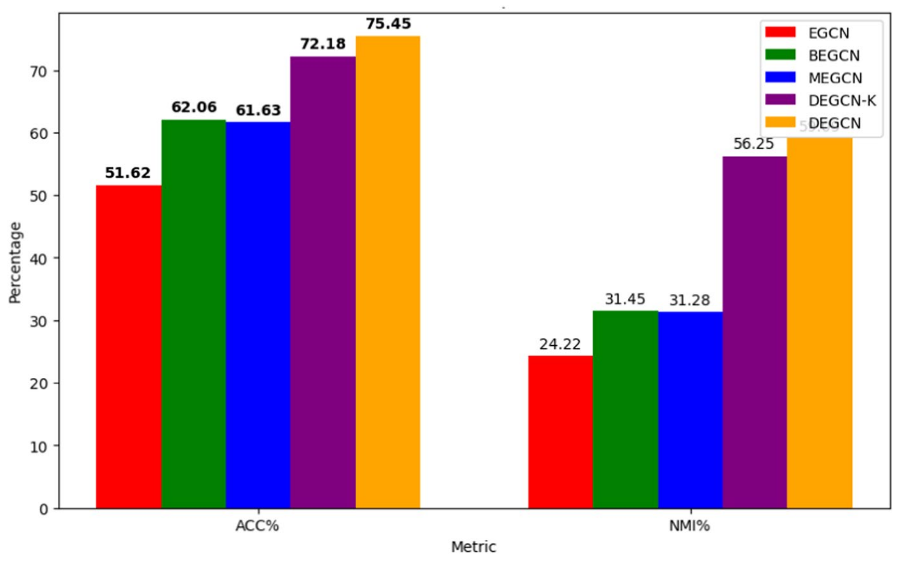

Fig. 7 Performance metric comparison for the CiteSeer dataset

6.2.1 Impact of Topology Module (BEGCN vs. EGCN)

Adding the topology module to the BEGCN model significantly improves the performance of the EGCN, which lacks it, across all datasets analyzed. In the Cora dataset, the topology module leads to a 4.6% rise in accuracy and a 6.03% rise in normalized mutual information (NMI). This shows how crucial structural knowledge is for making predictions better. Also, adding the topology module for the PubMed dataset makes it 5.26 percent more accurate and 6.99 percent better at NMI, which shows how useful it is for processing complicated biomedical data. The CiteSeer dataset shows the most pronounced improvements with the topology module, with a 10.44% increase in accuracy and a 7.23% increase in NMI. These improvements underscore the topology module's critical role in capturing and leveraging the structural nuances of the data, which contributes to a more robust and accurate performance across various types of graph data. Figure 8 shows the rise in ACC% and NMI% across the Cora, PubMed, and CiteSeer datasets after adding the topology module to the models.

6.2.2 Impact of the feature module (MEGCN vs. EGCN)

Incorporating the feature module into the MEGCN model results in significant performance improvements compared to the EGCN, which lacks this component, underscoring the feature module's efficacy in extracting and utilizing node-specific information. When the feature module is added to the Cora dataset, accuracy increases by 3.85%, and normalized mutual information (NMI) increases by 5.13%—demonstrating the module's ability to enhance model understanding through node attributes.

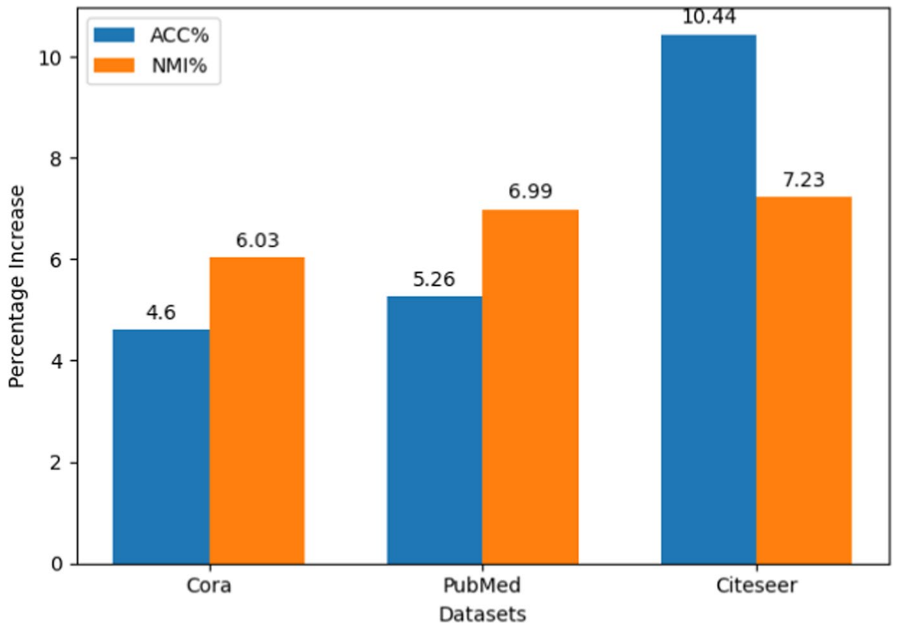

Fig. 8 Improvements in ACC% and NMI% with the topology module by dataset

The impact is even more significant in the PubMed dataset, with an increase in accuracy of 11.28% and an increase in NMI of 11.06%, highlighting the module's critical role in effectively handling detailed biomedical data. Similarly, in the CiteSeer dataset, adding the feature module leads to a 10.01% improvement in accuracy and a 7.06% increase in NMI. These improvements make it clear how important the feature module is for improving the model's performance by using attribute-rich data, which is necessary for getting more accurate results and better information synthesis across different datasets. Figure 9 shows the significant percentage improvements in ACC% and NMI% across the Cora, PubMed, and CiteSeer datasets attributed to adding the feature module to the models.

6.2.3 Influence of *k*-core preprocessing (DEGCN vs. DEGCN-K)

Including *K*-core preprocessing in the DEGCN model significantly enhances its performance over that of the DEGCN-K model, which does not utilize this preprocessing step, demonstrating the effectiveness of preprocessing in improving the model's focus and accuracy. In the Cora dataset, this preprocessing step leads to a 2.56% increase in accuracy and a 3.16% improvement in normalized mutual information (NMI). The benefits are similarly evident in the PubMed dataset, where the accuracy increases by 3.22%, and the NMI increases by 3.43%, underscoring the importance of refining the data representation and reducing noise. For the CiteSeer dataset, the improvements with *k*-core preprocessing are also notable, with a 3.27% increase in accuracy and a 2.84% increase in NMI. These results highlight the substantial

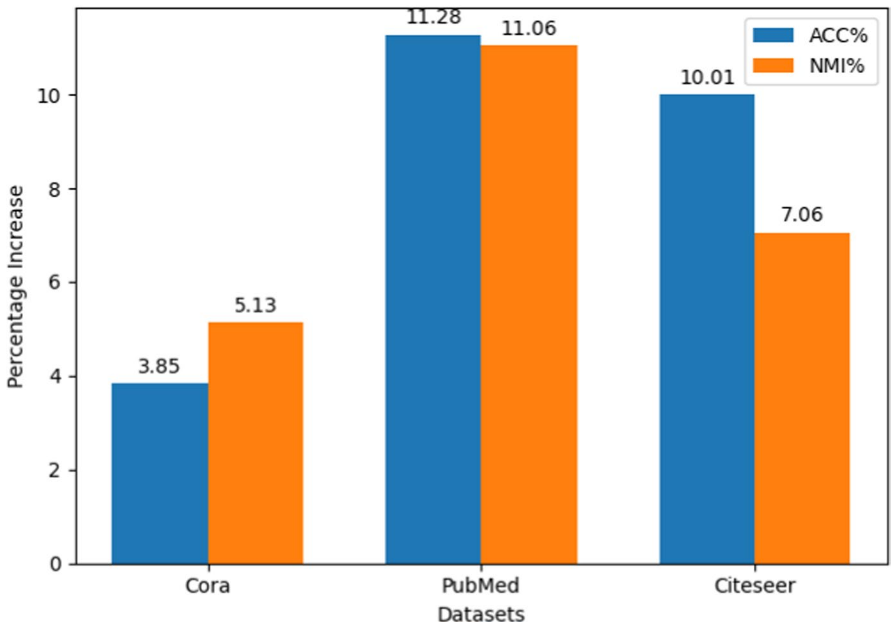

Fig. 9 Improvements in ACC% and NMI% with the feature module by dataset

impact of *K*-core preprocessing on the model's performance, particularly in enhancing the quality and centrality of the data processed, leading to more robust and accurate outcomes across different graph datasets. Figure 10 displays the higher accuracy (ACC%) and normalized mutual information (NMI%) achieved through *k*-core preprocessing on the Cora, PubMed, and CiteSeer datasets. This shows that it had a positive effect on model performance.

The analysis confirmed that each component of the DEGCN model contributes significantly to its performance across various metrics and datasets. The topology module is crucial for improving structural understanding, the feature module is critical in capturing essential node attributes, and *k*-core preprocessing enhances the model's focus and reduces noise. Each addition or enhancement in the model configuration leads to a marked improvement in performance, validating the integrated approach of the DEGCN model.

7 Graph visualization

We visualize the node representations of the Cora, PubMed, and CiteSeer datasets in two-dimensional space using *t*-distributed stochastic neighbor embedding (*t*-SNE) [49]. The results obtained from applying the t-SNE algorithm to three datasets, Cora, PubMed, and CiteSeer, are displayed in Figs. 11, 12, and 13, respectively. In each figure,

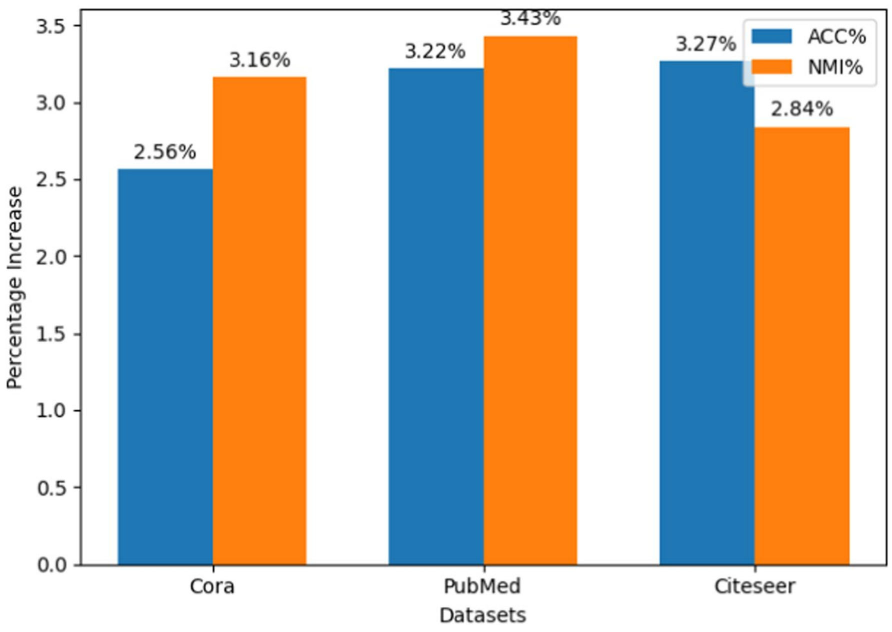

Fig. 10 Improvements in ACC% and NMI% with k-core preprocessing by dataset

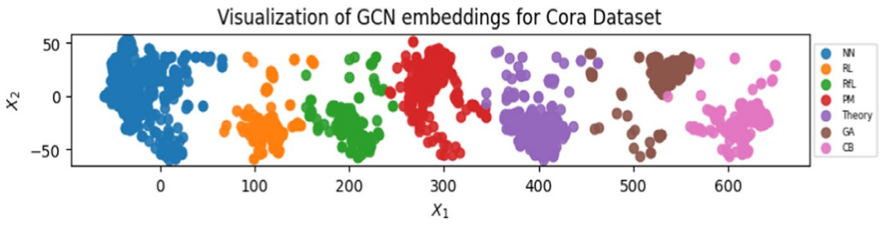

Fig. 11 2D visualization of node embeddings on the Cora dataset

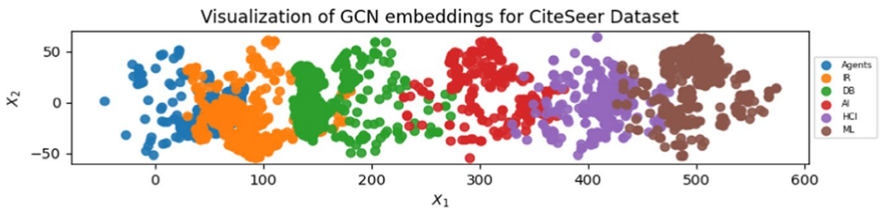

Fig. 12 2D visualization of node embeddings on the CiteSeer dataset

nodes within the same cluster are shown in the same color, while nodes in different clusters are depicted in various colors.

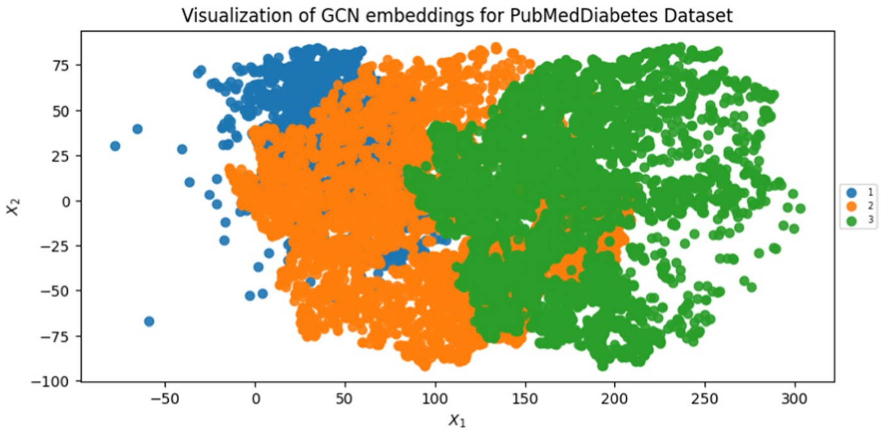

Fig. 13 2D visualization of node embeddings on the PubMed dataset

8 Conclusion and future work

With the rapid advances in information technology and artificial intelligence, social network analysis has attracted the attention of many researchers worldwide. Consequently, the role of social networks in today's digital life is not negligible. Among the current research trends in SNA, community detection is one of the most challenging tasks, with a high impact on performance and efficiency in large-scale networks. Considering this importance, we propose a novel method based on GNNs to address the community detection problem effectively and precisely. Using the DEGCN in the proposed architecture is a novel approach to community detection. The experimental results indicated that our proposed approach could surpass many state-of-the-art related works in the literature regarding accuracy, performance, and effort rate. In future work, we plan to extend our research by further exploring additional GNNs and their potential to enhance community detection capabilities. We aim to delve into more complex network structures and larger datasets to test the scalability and robustness of our proposed method.

Moreover, we will investigate incorporating multimodal data and applying our approach to heterogeneous networks. Another direction for future research is to improve the interpretability of the community detection process, making it easier for users to understand the reasoning behind the detected communities. Finally, we intend to explore real-world applications, particularly in detecting misinformation spread and influence maximization in social networks, to demonstrate the practical utility of our method.

Author contributions Omid Rashnodi involved in conceptualization, methodology, software, writing—original draft, visualization. Maryam Rastegarpour took part in conceptualization, methodology, validation, writing—original draft, writing—review and editing, supervision. Parham Moradi involved in conceptualization, validation, writing—review and editing. Azadeh Zamanifar took part in conceptualization, validation, writing—review and editing.

Data availability All datasets utilized in our study are publicly available and accessible online, ensuring that our work can be replicated and scrutinized by peers within the academic community. Specifically,

we have employed widely recognized datasets, including Cora, PubMed, and CiteSeer, as the foundation for our analysis. These datasets are critical resources in the fields of document classification, citation network analysis, and natural language processing, among others. For those interested in further exploration or replication of our study, the datasets can be found at their respective repositories: Cora and CiteSeer datasets are available through the "LINQS Datasets" website, and the PubMed dataset can be accessed via the "National Library of Medicine" website. By utilizing these publicly available resources, our research stands on a platform of transparency and reproducibility, core values that enhance the integrity and impact of our findings.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Wen X et al (2016) A maximal clique based multiobjective evolutionary algorithm for overlapping community detection. IEEE Trans Evol Comput 21(3):363–377
2. Lu X et al (2018) Adaptive modularity maximization via edge weighting scheme. Inf Sci 424:55–68
3. Wu W et al (2018) Nonnegative matrix factorization with mixed hypergraph regularization for community detection. Inf Sci 435:263–281
4. Altinoz OT, Deb K, Yilmaz AE (2018) Evaluation of the migrated solutions for distributing reference point-based multi-objective optimization algorithms. Inf Sci 467:750–765
5. Whang JJ, Gleich DF, Dhillon IS (2016) Overlapping community detection using neighborhood-inflated seed expansion. IEEE Trans Knowl Data Eng 28(5):1272–1284
6. Fortunato S, Hric D (2016) Community detection in networks: a user guide. Phys Rep 2016(659):1–44
7. Garza SE, Schaeffer SE (2019) Community detection with the label propagation algorithm: a survey. Physica A Stat Mech Appl 534:122058
8. Cao J et al (2018) Incorporating network structure with node contents for community detection on large networks using deep learning. Neurocomputing 297:71–81
9. He C et al (2019) Community detection method based on robust semi-supervised nonnegative matrix factorization. Phys A Stat Mech Appl 523:279–291
10. Chen Z, Li X, Bruna J (2020) Supervised community detection with line graph neural networks. International Conference on Learning Representations. p 1–24
11. Zhang T et al (2020) CommDGI: community detection oriented deep graph infomax. p 1843–1852
12. Tang J et al (2015) Line: large-scale information network embedding. The 24th International Conference on World Wide Web
13. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. p 855–864
14. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on KNOWLEDGE Discovery and Data Mining
15. Chen S, Guo W (2023) Auto-encoders in deep learning—a review with new perspectives. Mathematics 11(8):1777
16. Zhao S et al (2021) Hierarchical representation learning for attributed networks. IEEE Trans Knowl Data Eng 35(3):2641–2656
17. Lu H-Y et al (2024) Visual analytics of multivariate networks with representation learning and composite variable construction. IEEE Transactions on Visualization and Computer Graphics
18. Wang C et al (2017) Mgae: marginalized graph autoencoder for graph clustering. Conference on Information and Knowledge Management
19. Li B et al (2020) Multi-source information fusion based heterogeneous network embedding. Inf Sci 534:53–71
20. He C et al (2021) Boosting nonnegative matrix factorization based community detection with graph attention auto-encoder. IEEE Trans Big Data 8(4):968–981
21. Yang C et al (2021) Network embedding for graphs with node attributes, in network embedding: theories, methods, and applications. p 29–38

22. Zhang Y et al (2022) Spectral–spatial feature extraction with dual graph autoencoder for hyperspectral image clustering. IEEE Trans Circuits Syst Video Technol 32(12):8500–8511
23. Jin D et al (2021) A survey of community detection approaches: from statistical modeling to deep learning. IEEE Trans Knowl Data Eng 35(2):1149–1170
24. Liu F et al (2020) Deep learning for community detection: progress, challenges and opportunities. arXiv preprint arXiv:2005.08225
25. Zhou J et al (2020) Graph neural networks: a review of methods and applications. AI Open. p 57–81
26. Su X et al (2022) A comprehensive survey on community detection with deep learning. IEEE Trans Neural Netw Learn Syst
27. Jin D et al (2019) Graph Convolutional networks meet markov random fields: semi-supervised community detection in attribute networks. AAAI Conf Artif Intell 33(01):152–159
28. Sun H et al (2020) Network embedding for community detection in attributed networks. ACM Trans Knowl Discov Data (TKDD) 14(3):1–25
29. Jin D et al (2019) Community detection via joint graph convolutional network embedding in attribute network. In: International Conference on Artificial Neural Networks
30. Luo J, Du Y (2020) Detecting community structure and structural hole spanner simultaneously by using graph convolutional network based auto-encoder. Neurocomputing 410:138–150
31. Veličković P et al (2017) Graph attention networks. arXiv preprint arXiv:1710.10903
32. Goodfellow I et al (2020) Generative adversarial networks. Commun ACM 63(11):139–144
33. Chen H et al (2019) Exploiting centrality information with graph convolutions for network representation learning. International Conference on Data Engineering
34. Xin X et al (2017) Deep community detection in topologically incomplete networks. Phys A Stat Mech Appl 469:342–352
35. Cao S et al (2023) LGNN: a novel linear graph neural network algorithm. Front in Comput Neurosci 17:1288842
36. Zhang T et al (2020) CommDGI: community detection oriented deep graph infomax. International Conference on Information and Knowledge Management
37. Ruiqi H et al (2020) Going deep: graph convolutional ladder-shape networks. Proc AAAI Conf Artif Intell 34(03):2838–2845. https://doi.org/10.1609/aaai.v34i03.5673
38. Liu Y et al (2020) Independence promoted graph disentangled networks. Proc AAAI Conf Artif Intell 34(04):4916–4923. https://doi.org/10.1609/aaai.v34i04.5929
39. Levie R et al (2018) Cayleynets: graph convolutional neural networks with complex rational spectral filters. IEEE Trans Signal Process 67(1):97–109
40. Geisler S, Zügner D, Günnemann S (2020) Reliable graph neural networks via robust aggregation. Adv Neural Inf Process Syst 33:13272–13284
41. Cai X, Wang B (2023) A graph convolutional fusion model for community detection in multiplex networks. Data Min Knowl Disc 37(4):1518–1547
42. Li D, Zhang S, Ma X (2022) Dynamic module detection in temporal attributed networks of cancers. IEEE/ACM Trans Comput Biol Bioinf 19(4):2219–2230
43. Li D, Lin Q, Ma X (2021) Identification of dynamic community in temporal network via joint learning graph representation and nonnegative matrix factorization. Neurocomputing 435:77–90
44. Li D et al (2021) Detecting dynamic community by fusing network embedding and nonnegative matrix factorization. Knowl Based Syst 221:106961
45. Li D, Ma X, Gong M (2023) Joint learning of feature extraction and clustering for large-scale temporal networks. IEEE Trans Cybern 53(3):1653–1666
46. Huan H et al (2023) Diverse deep matrix factorization with hypergraph regularization for multi-view data representation. IEEE/CAA J Autom Sin
47. Huang H et al (2023) Exclusivity and consistency induced NMF for multi-view representation learning. Knowl-Based Syst 281:111020
48. Huang H et al (2024) Comprehensive multiview representation learning via deep autoencoder-like nonnegative matrix factorization. IEEE Trans Neural Netw Learn Syst. p 5953–5967
49. Amirfarhad Farhadi MM, Arash Sharifi, Mohammad Teshnelab (2024) Domaina daptation in reinforcement learning: a comprehensive and systematic study. Front Inf Technol Electron Eng
50. Kanatsoulis CI, Sidiropoulos ND, Claims AI (2022) GAGE: geometry preserving attributed graph embeddings. Fifteenth ACM International Conference on Web Search and Data Mining. p 439–448
51. Newman ME (2006) Modularity and community structure in networks. Proc Natl Acad Sci 103(23):8577–8582

52. Jianbo S, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 22(8):888–905
53. Liu L et al (2015) Community detection based on structure and content: a content propagation perspective. IEEE International Conference on Data Mining
54. Shchur O, Günnemann S (2019) Overlapping Community detection with graph neural networks
55. Zhang X et al (2019) Attributed graph clustering via adaptive graph convolution. arXiv preprint arXiv:1906.01210
56. Cui G et al (2020) Adaptive graph encoder for attributed graph embedding. p 976–985
57. Huang W (2021) Graph auto-encoders with edge reweighting. International Journal of Reconfigurable and Embedded Systems (IJRES)
58. Sen P et al (2008) Collective classification in network data. AI Mag 29(3):93–93
59. Namata G et al (2012) Query-driven active surveying for collective classification. In 10th International Workshop on Mining and Learning with Graphs
60. Rice SA (1927) The identification of blocs in small political bodies. Am Polit Sci Rev 21(3):619–627
61. Zhu W, Wang X, Cui P (2020) Deep learning for learning graph representations. In Deep learning: concepts and architectures. p 169–210
62. Cao S, Wei L, Qiongkai X (2016) Deep neural networks for learning graph representations. Proc AAAI Conf Artif Intell. https://doi.org/10.1609/aaai.v30i1.10179
63. Sun F-Y et al (2019) vGraph: a generative model for joint community detection and node representation learning. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. Curran Associates Inc
64. Tian F et al (2014) Learning deep representations for graph clustering. Proc AAAI Conf Artif Intell. https://doi.org/10.1609/aaai.v28i1.8916
65. Kipf T, Welling M (2016) Variational graph auto-encoders
66. Pan S et al (2019) Learning Graph Embedding With Adversarial Training Methods. IEEE Transactions on Cybernetics 50:1–13
67. Wang C et al (2019) Attributed graph clustering: a deep attentional embedding approach. pp 3670–3676
68. Zheng S et al (2020) Distribution-induced bidirectional generative adversarial network for graph representation learning. p 7222–7231
69. Park J et al (2019) Symmetric graph convolutional autoencoder for unsupervised graph representation learning
70. Guo L, Dai Q (2021) Graph clustering via variational graph embedding. Pattern Recogn 122:108334
71. Yang C et al (2015) Network representation learning with rich text information. In: Twenty-Fourth International Joint Conference On Artificial Intelligence
72. Xia R et al (2014) Robust multi-view spectral clustering via low-rank and sparse decomposition. Proceedings of the AAAI Conference on Artificial Intelligence; 28(1)
73. Ahmadi M, Safayani M, Mirzaei A (2022) Deep graph clustering via mutual information maximization and mixture model. arXiv preprint arXiv:2205.05168
74. Li J et al (2020) Dirichlet graph variational autoencoder
75. Xie H, Ning Y (2023) Community detection based on BernNet graph convolutional neural network. J Korean Phys Soc 83(5):386–395
76. Deng L, Guo B, Zheng W (2024) GCN-based weakly-supervised community detection with updated structure centres selection. Connect Sci 36(1):2291995
77. Ng A, Jordan M, Weiss Y (2002) On spectral clustering: analysis and an algorithm. Adv Neural Inf Process Syst
78. Tian F et al (2014) Learning deep representations for graph clustering. Proceedings of the National Conference on Artificial Intelligence. p 1293–1299
79. Sun F-Y et al (2019) vGraph: a generative model for joint community detection and node representation learning

Authors and Affiliations

Omid Rashnodi[1] · Maryam Rastegarpour[2] · Parham Moradi[3] · Azadeh Zamanifar[1]

✉ Maryam Rastegarpour
m.rastgarpour@gmail.com

Omid Rashnodi
omid.rashnodi@iau.ac.ir

Parham Moradi
p.moradi@uok.ac.ir

Azadeh Zamanifar
azamanifar@srbiau.ac.ir

[1] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

[2] Department of Computer, Saveh Branch, College of Engineering, Islamic Azad University, Saveh, Iran

[3] School of Engineering, RMIT University Melbourne, Melbourne, Australia