International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-025-02534-y

ORIGINAL ARTICLE

Local Community Detection Based on Core Nodes using Deep Feature Fusion

Xingjun Guo[1] · Xiaohong Li[1] · Wanyao Shi[1] · Siwei Wang[1]

Received: 31 May 2024 / Accepted: 2 January 2025

Abstract
Unlike global community detection, local community detection is to identify a cluster of nodes sharing similar feature information based on a given seed. The accuracy of many local community detection algorithms heavily relies on the quality of seed nodes. Only high-quality seed nodes can accurately detect local communities. At the same time, the inability to effectively obtain node attributes and structural information also leads to an increase in subgraph clustering error rates. This paper proposes a Local Community Detection based on Core Nodes using deep feature fusion, named LCDCN. We find the nearest nodes for the seed nodes, then construct a k-subgraph through a specific subgraph extractor based on the core nodes. Subsequently, two deep encoders are employed to encode and fuse the attribute and structure information of the subgraph, respectively. Finally, the local community is discovered by optimizing the fused feature representation through a self-supervised optimization function. Extensive experiments on 10 real and 4 synthetic datasets demonstrate that LCDCN outperforms its competitors in performance.

Keywords Local community detection · Clustering · Seed selection · Feature fusion

1 Introduction

Community detection aims to discover tightly connected sets of nodes within a network to understand better and interpret the hidden relationships in complex networks. Scholars have extensively discussed community detection methods in order to uncover the true community structure within complex networks more effectively. Existing methods primarily focus on identifying global community structures [1], and even overlapping community structures [2]. Global community detection is identifying all potential community structures within the network. However, real-world data often grows exponentially, making traditional global community detection methods inadequate for large-scale networks. In most cases, the focus is not on the global structure of large networks but rather on individual or small local structures near the seed nodes [3]. For example, identifying delinquent users within banking networks [4], discovering specific protein groups in biological fields [5], and providing targeted recommendations for products within recommendation systems [6]. Therefore, separating specific node sets into a local community [7] within a network enables the rapid analysis of the local composition of a large-scale network.

Local community detection is initiated from a specific seed nodes in the graph and identifies closely related communities to the seed nodes without traversing or analyzing the entire graph. In local detection methods [8], the majority of focus is placed solely on the local information of seed nodes without considering their effectiveness. Effective seed nodes are advantageous for detecting local communities; conversely, ineffective seed nodes can lead to significant discrepancies between community detection results and the actual node labels [9]. For example, when given seed nodes are located at the edge of a community or even at the global network edge, previous methods may fail to fully identify all community members to which the

X. Guo and X. Li contributed equally to this work.

✉ Xiaohong Li
xiaohongli@nwnu.edu.cn

Xingjun Guo
2022222279@nwnu.edu.cn

Wanyao Shi
2023222205@nwnu.edu.cn

Siwei Wang
2023212037@nwnu.edu.cn

[1] College of Computer Science and Engineering, Northwest Normal University, Lanzhou 730070, People's Republic of China

Published online: 23 January 2025

seed node belongs and may even incorrectly label them with erroneous community tags. This has a significant impact on local community discovery work. Therefore, the initialization of seed nodes directly affects the accuracy of local community discovery results. However, in conducting local community detection work [10], it is important to focus on the local information of any random node rather than the specific node's localized information. Thus, effectively finding the nearest and most important nodes locally by selecting randomly initialized seed nodes is crucial.

Although traditional approaches have made great strides in community detection, conventional algorithms for local community detection often yield suboptimal results that tend towards locally optimal outcomes. On the other hand, methods based on deep learning can explore more complete, comprehensive, and deeper complex information within networks addressing high-dimensional data problems [11] that traditional methods [12] cannot handle. While existing clustering methods have achieved the goal of community detection using deep learning techniques, they still face several unresolved challenges. Firstly, they independently obtain structure and attribute information, which cannot be perfectly fused [13]. Secondly, after multi-layer GNN networks, the obtained node features may be too similar to provide useful contributions to community detection [14]. Finally, These local clustering methods only focus on local single nodes but fail to pay attention to local structures. Therefore, they are not suitable for detection by the local community.

Based on these problems, we propose a new local community detection method based on core nodes, which not only provides an effective method for finding the optimal seeds but also enhances the effectiveness of local community detection by using deep learning technology. Specifically, we identify the true core nodes of the seed nodes by calculating the comprehensive scores of importance and correlation between the seed nodes and their *k*-neighbor nodes. This approach not only avoids the problem of invalid seed nodes, but also effectively finds the most influential core nodes, increasing the accuracy of local community discovery. Subsequently, we use *k*-subgraph extractors to focus on the subgraph information of nodes, rather than solely nodes attributes. This way transforms the node's information into more comprehensive and rich subgraph information. Finally, a attribute encoder and structural encoder are utilized to separately encode the attribute and structural information. The attribute information obtained at each layer is fed back into the corresponding layer's structural information, ensuring a more complete and integrated representation.

In conclusion, our main contributions to this paper are summarized as follows:

- We propose a new method to find the locally most important core nodes through random seed nodes, which effectively makes the community discovery results in a more accurate direction. In addition, we introduce a subgraph extractor to extract a more comprehensive subgraph structure from large-scale graphs to improve the efficiency of local community detection.
- We proficiently integrate the node attribute information obtained from the attribute encoder and the structural information obtained from the structure attention encoder, enhancing the node features for local clustering tasks. Additionally, we introduce a novel self-supervised optimization function to align the resulting local communities with real structures.
- We conduct comprehensive experiments on real and synthetic datasets to demonstrate the effectiveness of our approach.

2 Related work

Community detection, a prominent research focus in complex networks, has spurred the development of numerous algorithms in recent years. These methods typically detect communities by globally modifying the network topology. For example, the classic GN [15] algorithm detects communities by iteratively removing edges with the highest betweenness. Newman proposed the concept of modularity [16] based on the density of connections within communities and iterated execution to achieve community division. However, these approaches often face challenges such as local optima and limited scalability to large networks. To overcome these issues, the structural information theory-based method HSSInfo [17] adopts a parallel intersection and community merging algorithm, significantly reducing memory consumption and supporting high parallelism in large-scale networks scenarios. The effectiveness of this novel approach has been validated through its applications in SSSE [18] and UnDBot [19]. Unlike global community detection, local community detection aims to identify all nodes in the community where the seed nodes is located. Generally speaking, local community detection methods can be divided into traditional local community detection techniques and deep learning-based local graph clustering. In the following, we will introduce relevant work on local community detection from these two perspectives.

In traditional methods, the effective selection of seed nodes is crucial for accurately discovering the local community. Since the choice of seed nodes directly affects the detection results, local detection may produce different results from global algorithms. Therefore, the primary problem facing local community detection is selecting high-quality seeds. Many existing methods [20] ignore the importance of seed nodes, which will inevitably reduce the quality of the local community obtained under such a premise. HqsMLCD [21] reveals the overlapping communities in which the seed node is located by exploring the potential communities of seed nodes; MAPPR [22] selects a series of high-quality seed nodes with a target by minimizing the conductivity relationship. Although these methods alleviate the problem of "Seed Wasted" to some extent, only considering how to select or alternatively select from a single perspective cannot effectively solve the problem.

Another significant challenge in traditional methods for local detection is the effective expansion of the community. FLCS [23] introduces the concept of local modularity to expand the communities, and Shang et al. [24] improve local modularity to discover local communities. On the other hand, node similarity is also an effective indicator of the degree of connection between community members, so optimization methods [25] based on node similarity functions have also been used for local community expansion. LOCD [26] enhances the traditional similarity function and achieves the discovery of local communities by comparing fuzzy relations between nodes with greater centrality. What's more, ASFWF [27] employs alternating fusion strategies of strong and weak fusion for node fusion to expand local communities. JLDEC [28] dynamically decomposes the higher-order similarity matrix of the network to characterize its topology. It effectively and efficiently identifies the optimal local solution through parallel processing of three decomposition methods. It is important to note that these optimization extension methods focus solely on node or topological information without effectively fusing them. Moreover, while there is a greater emphasis on localization information, it is also essential to consider overall information about the localization network rather than individual nodes. Deep learning-based methods [29] can effectively address these problems that traditional methods cannot handle.

The potential of deep learning to capture information about the attributes and structure of a graph leads to a more comprehensive representation, making it better suited for downstream community detection tasks. NE2NMF [30] fuses network embedding and non-negative matrix factorization to detect dynamic communities. DAEGC [29] uses attention coefficients to represent the correlation between nodes and capture structural information within communities. However, this overlooks the importance of node attribute information. Recently proposed CDBNE [31] uses graph attention mechanisms when encoding topological structure and node attributes. JLMDC [32] utilizes network topology to extract node features guided by clustering, enabling the detection of communities within the network. However, there are redundancy issues with edge and node information encoded using the same attention mechanism. To address these issues, DDGAE [33] introduces a dual-view graph attention encoder that processes structural and attribute information as independent views to learn embeddings for nodes effectively but still faces problems related to similar information. AEGraph [34] obtains attribute representation based on structural information by enhancing attribute coding, which can be utilized for clustering tasks. SEA [35] acquires deep graph representation by strengthening the deep attention encoder of graph features, while DMGAE [36] introduces a deep variational autoencoder to learn node attribute information and enhance the accuracy of downstream clustering tasks. It is evident that applying deep learning technology to community detection, particularly local community detection, is a significant approach. Therefore, building upon the proposal of the optimal seed node, our method presents a deep graph encoder framework aimed at obtaining deeper and more comprehensive attribute and structural information for local community detection purposes.

Moreover, the random walk technique has proven effective in capturing intricate network information. RWI [37] calculates the random walk distance between nodes in the network. Nodes with significant influence on the community structure can be identified. Furthermore, AMQCS-RWS [38] utilizes random walk similarity to discover communities closely associated with query nodes and similar in the attribute space. TriDNR [39] adeptly integrated random walk and deep network representation to capture potential community structures by generating a sequence of nodes. Similarly, RoSANE [40] proposed a robust attribute network embedding method for sparse networks and further enhanced the embedding representation capability of sparse networks by leveraging random walk technology, thereby improving community detection performance.

3 Proposed method

In this section, we present the proposed LCDCN in detail, showing the overall framework shown in Fig. 1. We first find the core nodes most relevant to the given seed nodes. Next, we extract the k-subgraph of the core nodes as the model's input. Then, we encode the attribute and structure information of the k-subgraph. We connect each attribute encoder layer with the structure encoder's corresponding layer to fuse the attribute-specific representation into the structure-aware representation by a delivery operator. At the same time, we propose a joint loss to supervise the training of the attribute encoder and the structure encoder. Finally, we achieve the purpose of community detection. Below is a detailed description of our approach.

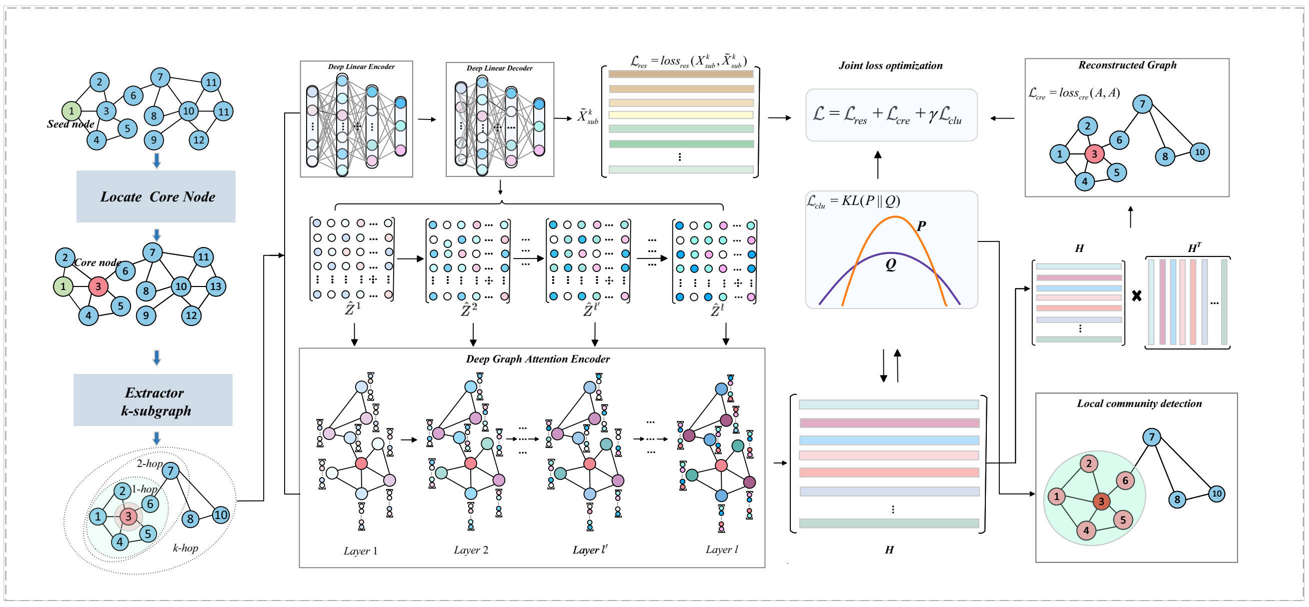

Seed node
Locate Core Node
Core node
Extractor k-subgraph
2-hop
1-hop
k-hop
Deep Linear Encoder
Deep Linear Decoder
$\mathcal{L}_{res} = loss_{res}(X^k_{sub}, \hat{X}^k_{sub})$
Joint loss optimization
$\mathcal{L} = \mathcal{L}_{res} + \mathcal{L}_{cre} + \gamma\mathcal{L}_{clu}$
Reconstructed Graph
$\mathcal{L}_{cre} = loss_{cre}(A, A)$
$\mathcal{L}_{clu} = KL(P \| Q)$
P
Q
Deep Graph Attention Encoder
Layer 1
Layer 2
Layer l'
Layer l
H
H^T
Local community detection

Fig. 1 Overall Framework of the LCDCN Model

3.1 Problem formulation and definition

In the following, we refer to represent an undirected and unweighted graph with $|V| = N$ nodes as $G = (V, E, \mathbf{X})$, where $V = \{v_1, v_2, ..., v_N\}$ denote a set of nodes, $E = \{e_{vu} | \forall v, u \in V\}$ denotes a set of edges between v and u. The node attributes for $u \in V$ are denoted by $\mathbf{x}_u \in \mathbb{R}^d$, so the node attributes for all nodes are stored in $\mathbf{X} \in \mathbb{R}^{N \times d}$ for a graph with N nodes. Also, d is the dimension for node attributes. Similarly, we let $G^k_{sub} = (V^k_{sub}, E^k_{sub}, \mathbf{X}^k_{sub})$ denote a subgraph with $\left|V^k_{sub}\right| = N^k_{sub}$ nodes of G, where $V^k_{sub} = \{v_1, v_2, ..., v_{N^k_{sub}}\}$ denote a set of nodes, $E^k_{sub} = \{e_{vu} | \forall v, u \in V^k_{sub}\}$ denotes a set of edges between v and u. And the subgraph node attributes for $v \in V^k_{sub}$ is also denoted by $\mathbf{x}_v \in \mathbb{R}^d$, so the node attributes for all subgraph are stored in $\mathbf{X}^k_{sub} \in \mathbb{R}^{N^k_{sub} \times d}$. Given a seed node v, our goal is to detect the set of nodes most similar to the seed node.

3.2 Location of core nodes

For local community detection, the seed node's quality directly impacts the community's quality. Therefore, identifying the seed node is crucial in detecting the local community. A new method is proposed to locate the core nodes that exert greater influence on the regional structure in random nodes. First, randomly select a node v as the initial seed node and calculate the importance of all nodes within its second-order neighbors using Eq. (1). It is widely recognized that the degree of a node directly reflects its importance. Equation (1) also utilizes the initial seed's first two layers of neighbor information. Compared to the importance function that only uses one layer of neighbors, $Centrality(v)$ can better obtain the importance of the initial seed node.

$$Centrality(v) = deg(v) + \sum_{u \in N(v)} \left(\lambda_1 deg(u) + (1 - \lambda_1) \sum_{w \in N(u)} deg(w) \right) \tag{1}$$

here, $deg(v)$ is the degree of the node, $N(v)$ is the neighbor set of node v, and λ_1 is a parameter that regulates the importance of neighboring nodes. After calculating the importance of the k-neighbor nodes of v respectively, we will select the top κ most important nodes and calculate the correlation between them and the random initial seed node v, as shown in Eq. (2).

$$Rele(v, u) = \lambda_2 f_1(v, u) + (1 - \lambda_2) f_2(v, u) \tag{2}$$

$$f_1(v, u) = \frac{|N(v) \bigcap N(u)|}{|N(v) \bigcup N(u)|} \tag{3}$$

$$f_2(v, u) = \frac{|N(N(v)) \bigcap N(N(u))|}{|N(N(v)) \bigcup N(N(u))|} \tag{4}$$

where the $Rele(v, u)$ represents the correlation between nodes v and u, and $f_1(v, u)$ denotes the first-order correlation between nodes v and u. Similarly, $f_2(u, v)$ denotes the second-order correlation between nodes v and u, and λ_2 is a parameter that controls the importance of first and second-order correlations. Finally, we calculate the Nearest with Greater Significance and Correlation(NGSC) score of each node in the k-order neighbors of the seed node v. The NGSC score is obtained by summing the importance and relevance of each node. Then, We select the node with the highest score as the local NGSC node of the initial seed nodes, i.e., the core nodes.

$$S_{NGSC}(u) = \alpha Centrality(u) + \beta Rele(v, u) \tag{5}$$

where, $S_{NGSC}(u)$ represents the NGSC score of node u. α and β represent the importance and relevance of joint coefficients, respectively. Empirically, to make the score more realistic, we set α=0.1 and β=1.

3.3 Extraction of *k*-subgraph

To efficiently perform local community detection on large-scale graphs using deep learning methods, we extract the k-subgraph G^k_{sub} based on k-neighbors of the core nodes u as the input for local communities. While directly accessing the k-subtree [41] of core nodes u has high computational efficiency, it limits the transmission of node information, which is closely related to node feature aggregation. Influenced by [42], to acquire nodes in the subgraph more globally, we use a GNN network to extract u's k-hop neighborhood of nodes in the k-subgraph to update u's representation instead of simply using node u itself. Finally, we use a summation pooling function to aggregate k-hop neighborhood information. Therefore, the k-subgraph extractor updates nodes u within the k-subgraph, as shown in Eq. (6).

$$Extra(u, G^k_{sub}) = \sum_{v \in N_k(u)} GNN_{G^k_{sub}}(v) \tag{6}$$

where, $Extra(u, G^k_{sub})$ represents the k-hop node feature extractor, the k denotes the number of hops. $N_k(u)$ refers to the set of nodes within k hops of node u, while G^k_{sub} is the k-subgraph extracted based on the global graph with core nodes as a foundation. Similar to GNN, we enhance the expressiveness of the extractor by initializing original graph features $\mathbf{X}$ as inputs for the k-subgraph extractor. Finally, the aggregate feature representation of the k-subgraph is obtained through the k-subgraph extractor, which is denoted as $\mathbf{X}^k_{sub}$. Specifically, the feature of node u is represented as $\mathbf{x}(u)^k_{sub}$. Despite continuously obtaining k-hop information for nodes through the k-subgraph extractor, its local subgraph G^k_{sub} has lightweight characteristics regarding node count and hop number. This approach not only avoids additional overhead but also maximizes performance efficiency.

3.4 Deep linear AutoEncoder

Node attribute information is important, and effectively obtaining the node's attributes is essential for discovering more appropriate communities. Therefore, we propose a Deep Linear AutoEncoder (DLAE) to encode the node's local attribute information for generality. Particularly, through the ℓ-layer encoder, the nodes with different characteristics are encoded from the original k-subgraph to enrich the hierarchical attribute information of nodes. Moreover, the encoded attribute information by ℓ fully connected layers is fed into the ℓ -layer attention network to obtain a more appropriate node representation.

We assume that the DLAE has ℓ layers of neural networks. Specifically, the output $\hat{\mathbf{z}}$ of the ℓ-th layer of the autoencoder can be denoted as Eq. (7).

$$\hat{\mathbf{z}}^{\ell}_{v} = \delta\left(\mathbf{w}^{\ell}_{e}\hat{\mathbf{z}}^{\ell-1}_{v} + b^{\ell}_{e}\right) \tag{7}$$

where $\delta(.)$ is a non-linear activation function, such as ReLU or Tanh, and $\mathbf{w}^{\ell}_{e}$ and b^{ℓ}_{e} are the weight matrix and bias of the ℓ-th layer for encoder.$\hat{\mathbf{z}}^{\ell-1}_{v}$ is the attribute information representations in the ℓ-1 layer. Specifically, the input of the 0-th layer of the deep linear autoencoder is the node feature $\mathbf{X}^k_{sub}$ of the k-subgraph, and the output of the ℓ-th layer is $\hat{\mathbf{Z}}^{\ell}$, which serves as the final output of the encoder. In the same way, we define a similar decoder based on the encoder function as Eq. (8). The decoder aims to map the encoded result back to the original space, thereby reconstructing the generated data.

$$\mathbf{z}^{\ell}_{v} = \delta\left(\mathbf{w}^{\ell}_{d}\mathbf{z}^{\ell-1}_{v} + b^{\ell}_{d}\right) \tag{8}$$

here, $\mathbf{w}^{\ell}_{d}$ and b^{ℓ}_{d} are the weight matrix and bias of the ℓ-th layer for the decoder. Correspondingly, the input of the 0-th layer of the decoder is the output $\hat{\mathbf{Z}}^{\ell}$ of ℓ-th layer of the encoder, and the output of the ℓ-th layer is $\tilde{\mathbf{X}}^k_{sub}$, which is the final output of the decoder.

Considering that the output of the decoder is a reconstruction of the original graph, we formulate a loss function as Eq. (9) with the goal of minimizing the disparity between the decoder output and the original data. Training deep linear autoencoder in this manner can more effectively enhance the attribute information of nodes.

$$\begin{aligned}\mathcal{L}_{res} &= loss_{res}(\mathbf{X}^k_{sub}, \tilde{\mathbf{X}}^k_{sub}) \\ &= \frac{1}{2N^k_{sub}} \sum_{i=1}^{N^k_{sub}} \| \mathbf{x}_i - \tilde{\mathbf{x}}_i \|^2_2 = \frac{1}{2N^k_{sub}} \| \mathbf{x}^k_{sub} - \tilde{\mathbf{x}}^k_{sub} \|^2_2\end{aligned} \tag{9}$$

where $\mathcal{L}_{res}$ represents the attribute loss function, $\mathbf{x}^k_{sub}$ denotes the feature representation obtained after fusion by the k-subgraph extractor, $\tilde{\mathbf{X}}^k_{sub}$ is the output feature representation of the decoder, N^k_{sub} stands for the number of nodes in the k-subgraph, and $|| \cdot ||^2_2$ signifies the square of the L_2 norm.

3.5 Graph attention AutoEncoder

In this section, we encode the local structural information of the k-subgraph using a novel Graph Attention AutoEncoder(GAAE) compared to previously. At the same time, the GAAE simultaneously incorporates the node attribute information learned in the DLAE to synthetically process two types of enhanced feature representations, thereby improving the accuracy of community detection results. Consequently, we obtain a comprehensive representation of node information through the ℓ-layer GAAE.

3.5.1 Encoded of k-subgraph by graph Attention

To learn more comprehensive node structure information, we constructed a variant of the graph attention network for a graph encoder to learn the attribute and structure information of the subgraph. Specifically, the encoder learns the embedding representation of nodes by encoding various neighbor information for each node. Subsequently, considering the limitations of the traditional coefficient of concern, which only considers the absolute coding between graphs and disregards the measurement of structural similarity, we have opted to utilize the new structural similarity as the coefficient of concern. This is intended to gauge the significance of different nodes from a graph structure perspective. Therefore, different neighbors will have varying levels of importance, which aligns with the characteristics of the attention network and results in node features with different emphases. Thus, we represent the embedding of node v using a layer-wise graph attention strategy, as shown in Eq. (10).

$$\mathbf{h}_v^{\ell} = \sum_{u \in N(v)} \frac{p_e(\mathbf{h}_v^{\ell-1}, \mathbf{h}_u^{\ell-1})}{\sum_{w \in N(v)} p_e(\mathbf{h}_v^{\ell-1}, \mathbf{h}_w^{\ell-1})} f(\mathbf{h}_u^{\ell-1}) \tag{10}$$

where $f(\mathbf{x}) = \mathbf{W}_f \mathbf{x}$ is the linear value function and $\mathbf{W}_f$ is a linear projection matrix that can be trained.P_e represents a novel attention coefficient that is parameterized by trainable linear projection matrices $\mathbf{W}_\alpha$ and $\mathbf{W}_\alpha$, as shown in Eq. (11).

$$p_e(\mathbf{h}_v, \mathbf{h}_u) = \exp\left(\frac{\mathbf{W}_\alpha \mathbf{h}_v \cdot \mathbf{W}_\beta \mathbf{h}_u}{\left\|\mathbf{W}_\alpha \mathbf{h}_v\right\|_2^2 + \left\|\mathbf{w}_\beta \mathbf{h}_u\right\|_2^2}\right) \tag{11}$$

It is noteworthy that we designate the initial 0-th layer feature input of GAAE as the output $\mathbf{X}_{sub}^k$ of the k-subgraph extractor, as formally described in Eq. (12).

$$\mathbf{h}_v^1 = \sum_{u \in N(v)} \frac{p_e(\mathbf{x}(u)_{sub}^k, \mathbf{x}(u)_{sub}^k)}{\sum_{w \in N(v)} p_e(\mathbf{x}(v)_{sub}^k, \mathbf{x}(w)_{sub}^k)} f(\mathbf{x}(u)_{sub}^k) \tag{12}$$

Topologically, edges are the only indicators of connections between nodes, and the number of common neighbors indicates the strength of node connections. While the attention coefficient structurally focuses on the correlation between nodes, it does not account for the edge strength. Therefore, a new calculation method is being considered to measure the strength of connections between nodes and incorporate it into the attention network as an attention weight. Since the graph exhibits complex structural relationships, we calculate the connection strength by determining the number of common neighbors of node neighbors, as shown in Eq. (13).

$$p_f(v, u) = \sum_{x \in N(v)} \sum_{y \in N(u)} \delta(x, y) \tag{13}$$

here, $p_f(v, u)$ is the fiend-measure where $\delta(v, u) = 1$ if $e_{vu} \in E$ and $\delta(v, u) = 0$ otherwise. Therefore, Eq. (10) is rewritten as Eq. (14) with a fiend-measure value.

$$\mathbf{h}_v^{\ell} = \sum_{u \in N(v)} \frac{p_e(\mathbf{h}_v^{\ell-1}, \mathbf{h}_u^{\ell-1}) p_f(v, u)}{\sum_{w \in N(u)} p_e(\mathbf{h}_v^{\ell-1}, \mathbf{h}_w^{\ell-1})} f(\mathbf{h}_u^{\ell-1}) \tag{14}$$

The attention encoder pays more attention to the structural information of the graph and less attention to the attribute information, resulting in insufficient expression of node features. Therefore, we utilize the attribute information obtained from the DLAE as input for the GAAE encoder to compensate for the missing attribute information. Specifically, we consider combining the attribute information representations $\mathbf{z}_v^{\ell}$ from the DLAE with $\mathbf{h}_v^{\ell}$ from the GAAE encoder for enhanced feature representations as Eq. (15).

$$\tilde{\mathbf{h}}_v^{\ell} = (1 - \mu)\mathbf{z}_v^{\ell} + \mu \mathbf{h}_v^{\ell} \tag{15}$$

where the trade-off parameter μ balances the impacts of attribute information and structure information. In this way, the GAAE encoder can pay attention to structural and attribute information at each layer. Therefore, we use $\mathbf{z}_v^{\ell}$ as the input for the ℓ-1-th layer attention network to generate the node representation of the ℓ-th layer.

$$\mathbf{h}_v^{\ell} = \sum_{u \in N(v)} \frac{p_e(\mathbf{h}_v^{\ell-1}, \mathbf{h}_u^{\ell-1}) p_f(v, u)}{\sum_{w \in N(v)} p_e(\mathbf{h}_v^{\ell-1}, \mathbf{h}_w^{\ell-1})} f(\tilde{\mathbf{h}}_u^{\ell-1}) \tag{16}$$

3.5.2 Decoded of k-subgraph by graph attention

We construct the loss function using the simplest graph decoder to optimize the output representation of the encoder. Different decoders focus on various aspects of information within the graph. We reconstruct the subgraph using the output of the encoder and then create a loss function based on the reconstructed and original graphs.

Inspired by the previous work of [43], nodes with higher similarity are more likely to be connected by edges in the feature space. The similarity between nodes u and v can be approximatively represented by the inner product of their features $\mathbf{h}_u$ and $\mathbf{h}_v$. Consequently, a larger inner product indicates a higher likelihood of interconnected edges between the nodes. Finally, by applying the sigmoid function, these inner products are transformed into values within the range (0, 1), allowing for estimation of the probability of adjacency between nodes. This process generates a reconstructed adjacency matrix $\widetilde{\mathbf{A}}$ similar to the A. So, we adopt the inner product of the GAAE encoder's output $\mathbf{H}$ and its transpose to predict the connection relationship between nodes in the reconstructed graph. We reconstruct the adjacency matrix as Eq. (17).

$$\widetilde{\mathbf{A}} = \sigma(\mathbf{H}^{\mathrm{T}}\mathbf{H}) \tag{17}$$

where $\widetilde{\mathbf{A}}$ is the reconstructed structure matrix of the k-subgraph and σ is the activation function, such as sigmoid.

3.5.3 Loss function of graph attention

We reconstruct the original graph using the above method to minimize the difference between the reconstructed image and the original graph. The encoder is optimized to generate more appropriate node features by minimizing the Eq. (18).

$$\begin{aligned}\mathcal{L}_{cre} &= loss_{cre}(\mathbf{A}, \tilde{\mathbf{A}}) \\ &= -\frac{1}{N_{sub}^k}\sum_{i=1}^{N_{sub}^k}\sum_{j=1}^{N_{sub}^k}(a_{ij}\log \tilde{a}_{ij} + (1-a_{ij})\log(1-\tilde{a}_{ij}))\end{aligned} \tag{18}$$

3.6 Joint self-supervised optimization

While high-quality embeddings accurately capture node attribute and structure information, they may not always result in the most optimal community detection. Local community detection aims to identify a group of similar nodes based on a seed node, which must be integrated as a step in embedding optimization. This is crucial and essential for the task of community detection. On the other hand, node labels are not considered when learning node attributes and features, which is detrimental to community detection. To align the generated node embedding representation more closely with the characteristics of community detection while aggregating structure and attributes, we propose an optimization strategy for embedding. The aim is to enhance the usefulness of the generated node representation for community detection.

We evaluate this score based on similarity to find the set of remaining nodes that belong to the same community as the seed node. The higher the similarity, the more likely it is that the seed node will become a member of the community. Therefore, we first calculate the similarity $coS(v, u)$ between the seed node v and the remaining local nodes u using cosine similarity, as shown in Eq. (19).

$$coS(v,u) = \frac{\mathbf{h}_v \cdot \mathbf{h}_u}{\| \mathbf{h}_v \|_2^2 \cdot \| \mathbf{h}_u \|_2^2} \tag{19}$$

To comprehensively measure the symmetric difference between the seed node and surrounding nodes, we propose a new loss function that considers the similarity between nodes to minimize the difference between them, thereby making similar nodes more similar and dissimilar nodes less so. The nodes are more dissimilar, which helps to find more accurate communities based on seed nodes. Therefore, self-supervised community optimization is performed on node embeddings according to Eq. (20)

$$\mathcal{L}_{clu} = loss_{clu}(P, Q) = KL(P\|Q) \tag{20}$$

If P and Q are significantly different from each other, they will have no overlap, resulting in the disappearance of the gradient. Therefore, we modified it and proposed $S = \frac{(P+Q)}{2}$ so that the original asymmetric loss becomes a symmetric structure, as shown in Eq. (21).

$$\begin{aligned}\mathcal{L}_{clu} &= loss_{clu}(P, Q) = \frac{1}{2}KL(P\|S) + \frac{1}{2}KL(Q\|S) \\ &= \frac{1}{2}(\sum_{u=1}^{N_{sub}^k} p_{uv}\log\frac{p_{uv}}{s_{uv}} + \sum_{u=1}^{N_{sub}^k} q_{uv}\log\frac{q_{uv}}{s_{uv}})\end{aligned} \tag{21}$$

the q_{uv} measures the similarity between node embedding u and community seed node embedding v, and we measure it using the Eq. (21) distribution. Specifically, q_{uv} represents the soft label distribution indicating whether each node and the community seed node belong to the same community.

$$q_{uv} = (1 + (coS(u,v))^{-1})^{-1} \tag{22}$$

After obtaining the community labels for both the nodes and seed nodes, our objective is to facilitate the gathering of nodes similar to the seed node while ensuring that dissimilar nodes remain distant from the seed node. To achieve this, we establish that the label distribution's target distribution is p_{uv}, as shown in Eq. (23).

$$p_{uv} = \frac{q_{uv}^2}{\sum_{i=1}^{N_{sub}^k} q_{uv}} \tag{23}$$

We supervise the current community distribution through the target distribution Q, making the resulting embeddings easier to find communities based on seed nodes. In this way, we combine the structural and attribute loss functions to construct a new joint target loss function, such as Eq. (24).

$$\mathcal{L} = \mathcal{L}_{res} + \mathcal{L}_{cre} + \gamma \mathcal{L}_{clu} \tag{24}$$

where $\mathcal{L}_{res}$ and $\mathcal{L}_{cre}$ are the attribute loss function and the structural loss function, and $\mathcal{L}_{clu}$ is the community optimization loss function. The hyper-parameter γ is a coefficient that controls the balance in between.

As in the traditional, we jointly optimize the loss function through stochastic gradient descent to achieve optimal parameters. When the network is trained until the model converges, we take the result of the soft assignment Q optimization loss as the final community detection result. Formally, as shown in Eq. (25).

$$\hat{y}_u = \arg\max q_{uv} \tag{25}$$

4 Experiments

In this section, we conducted comprehensive experiments to demonstrate the superiority of the proposed LCDCN framework in local community detection on widely used graph networks.

4.1 Datasets

4.1.1 Real-world networks

We conducted experiments on ten standard real-world networks of various scales commonly used for evaluating attributed graph analysis. These networks consist of seven large-scale datasets and three relatively small-scale datasets. Detailed statistical information about the datasets is presented in Table 1. Specifically, The Polblogs dataset [44] focuses on political blogs in the United States, categorizing each blog as conservative or liberal based on its political attributes. In contrast, the Mich dataset [45] is sourced from a university's social network and obtained from Facebook. The DBLP [46] dataset contains comprehensive information about academic papers, including abstracts, authors, year of publication, venue, and title. The Orkut [47] dataset includes friendship connections and real communities from the Orkut.com online platform. The BlogCatalog [46] is a social blog relationship network that contains social relationships between bloggers and group members. Wikipedia [46] provides an English dataset with annotations specifically designed for domain detection sourced from Wikipedia articles. The HepPh [48] citation network is derived from the arXiv and features nodes representing papers and edges representing citation relationships. The Slashdot [48] social network comprises users as nodes and their friendships as edges. In the WebStanford [48] dataset, each node represents a page in a web network, and an edge signifies a connection between two different pages. The Amazon [49] network is a product purchase network where connections link two products purchased together, and communities represent various product categories. These datasets are commonly used to evaluate methods for embedding attributed graphs. In all datasets, each node belongs to one community. We divide the dataset randomly into 60% for training, 20% for testing, and 20% for validation purposes.

4.1.2 Synthetic networks

We selected real datasets and artificial synthetic networks generated based on the artificial generative model LFR [50]. The model method in this article can be more comprehensively evaluated through such a unique dataset. Therefore, according to the control parameters and their meanings in Table 2, we generated five special datasets of varying scales through LFR, named LFR-0, LFR-1, LFR-2, LFR-3, and LFR-4 in Table 3, respectively. Particularly, to ensure that the visual experiments in the following text can visually reflect the accuracy of the algorithm and the differences between the baseline algorithm, we only choose to use LFR-0 with a small number of nodes for visual experiments and use the rest of the dataset for other experiments.

Table 1 Real-world dataset summary

Dataset	Nodes	Edges	Communities
Polblogs	1490	19090	2
Mich	2933	54903	13
Dblp	12547	55748	4
Orkut	11751	237171	5
BlogCatalog	10312	333983	39
Wikipedia	4777	184812	40
HepPh	34546	421578	Unknown
Slashdot	77360	905468	Unknown
WebStanford	281903	2312497	Unknown
Amazon	334863	925872	49732

Table 2 Parameters for the LFR benchmark

Parameter	Description
N	number of nodes
$\overline{k}$	average degree
$maxdeg$	maximum degree
ϵ	mixing parameter
τ_1	minus exponent for the degree sequence
τ_2	minus exponent for the community size distribution
$minc$	minimum for the community sizes
$maxc$	maximum for the community sizes

Table 3 Parameters configuration of LFR-0, LFR-1, LFR-2, LFR-3, and LFR-4

Networks	N	$\bar{k}$	$maxdeg$	ϵ	τ_1	τ_2	$minc$	$maxc$
LFR-0	100	5	15	0.1	2	1	10	50
LFR-1	1000	10	50	0.1	2	1	20	50
LFR-2	5000	10	50	0.1	2	1	20	90
LFR-3	10000	10	50	0.1	2	1	20	100
LFR-4	20000	10	50	0.1	2	1	20	50

4.2 Experimental settings

4.2.1 Evaluation metrics

To comprehensively evaluate our proposed method, we employed three evaluation metrics, which are widely recognized in local community detection. The first metric is the F_1 score, a well-known measure that combines *Recall* and *Precision*, and is defined as follows:

$$F_1(\mathcal{C}_{det}, \mathcal{C}_{ture}) = 2 \times \frac{Precision(\mathcal{C}_{det}, \mathcal{C}_{ture}) \cdot Recall(\mathcal{C}_{det}, \mathcal{C}_{ture})}{Precision(\mathcal{C}_{det}, \mathcal{C}_{ture}) + Recall(\mathcal{C}_{det}, \mathcal{C}_{ture})} \tag{26}$$

In general, the F_1 score ranges from [0, 1], where a higher value indicates improved precision in community detection achieved by our algorithm.

Furthermore, we also use another commonly used metric for local community detection, called Conductance *Con*, to evaluate the accuracy of our algorithm. The *Con* refers to the ratio of internal nodes to edges connected to the outside of the local community, which can accurately describe the tightness of the internal structure of the local community. The definition of *Con* is as follows:

$$Con(C) = \frac{\sum_{u \in C} \sum_{v \in \overline{C}} \varphi(u, v)}{min(vol(C), vol(\overline{C}))} \tag{27}$$

When $\varphi(u, v) = 1$, node u and v are connected through an edge. Otherwise, $\varphi(u, v) = 0$. In addition,$vol(C) = \sum_{u \in C} deg(u)$, and the other end point in the complement set $\overline{C} = V \backslash C$. Correspondingly, the value of *Con* also falls within the [0, 1] range. On the contrary, the difference from the F_1-score is that the lower [51] the *Con*, the better the accuracy of the community detection results by our algorithm will be.

Finally, we choose a third evaluation metric, the local NMI evaluation index, to further reveal the community structure information. Different from the global NMI value, the calculation of local NMI only focuses on the local community structure where the seed node is located, without considering the overall community structure, so that the calculation results can more accurately reflect the accuracy of the local community. The NMI_{Local} value is defined as follows:

$$NMI_{Local} = -2 \times \frac{\sum_i \sum_j X_{ij} \log\left(\frac{X_{ij} N}{X_i X_j}\right)}{\sum_i X_i \log\left(\frac{X_i}{N}\right) + \sum_j X_j \log\left(\frac{X_j}{N}\right)} \tag{28}$$

Similar to the global *NMI*, when NMI_{Local} is close to 1, the detected community structure is closer to the real situation; on the contrary, it means that the detected community quality is low.

4.2.2 Baseline algorithms

To verify the performance of the proposed architecture, we selected several state-of-the-art methods recently proposed based on community detection to conduct comparative experiments with the LCDCN. The baseline methods include recent state-of-the-art deep community detection methods like EFR-DGC, DFCN, and SDCN. In addition, considering that traditional methods dominate the field of local community detection, we have selected some recent traditional methods that have demonstrated superior performance in this area for comparison. Such as ASFWF, LCDS-A, LCDPC, and LCDMD. The summary of all compared methods is as follows:

- (**EL-Trans** [52]) The EL-Trans is a deep learning mode in local community detection; the proposed knowledge graph Embedding model is based on entity feature information and local importance.
- (**SDCN** [53]) The SDCN algorithm uses a novel transfer operator and dual self-supervision modules. In this way, multiple data structures, from low-order to high-order, are naturally combined with the various representations learned by the autoencoder.
- (**DFCN** [13]) The DFCN integrates two sample embeddings from the local and global levels to perform consensus representation learning. Afterward, a more accurate target distribution is obtained by estimating the similarity between sample points and precomputed cluster centers in the latent embedding space.
- (**ASFWF** [27]) The ASFWF algorithm employs two strategies: strong fusion, which takes into account the information and connections between the two nodes and the local community, and weak fusion, which evaluates

parameter similarity. These strategies are alternately cycled to fuse the nodes.

- (**LCDS-A** [54]) The LCDS-A determines the most stable anchor community by tracking the evolution of key nodes in the community over time, which are referred to as anchors.
- (**LCDPC** [55]) The LCDPC first replaces the given node with a new seed node by calculating node importance and node similarity. Finally, the similar nodes between the potential community and the extended community of qualified nodes are selected and added to the initial community to expand the community.
- (**LCDMD** [56]) The LCDMD algorithm is divided into two phases: the core area detection phase and the local community expansion phase, based on the proximity of the local community. Finally, these two different stages are alternately performed to detect local communities that satisfy the specified conditions.

4.2.3 Experimental settings

To ensure the fairness of the experiment and enable the baseline methods to achieve the best performance, we adhered to the parameter settings specified in the original paper of the baseline experiment. For our model, we set the number of layers ℓ of the k-subgraph deep linear autoencoder with $L = 5$ fully-connected layers. At the same time, we maintain the number of layers ℓ of k-subgraph attention autoencoder with the same fully connected layer, resulting in the k-subgraph attention auto-encoder layer $L = 5$. At the beginning of the experiment, we initially employed an end-to-end approach to pre-train the deep linear autoencoder and attention autoencoder. This approach serves a dual purpose: initializing the model parameters in the network and accelerating the model's convergence to stability. Specifically, we set different dimensions in each layer of the deep linear autoencoder and attention autoencoder but ensure that the deep linear autoencoder and attention autoencoder have the same output dimension at each layer. Specifically, we set $Hidden\_1 = 1024$, $Hidden\_2 = 512$, $Hidden\_3 = 256$, $Hidden\_4 = 128$, $Hidden\_5 = 32$, and $Embedding = 16$. Based on experience, during the pre-training stage, we utilize the commonly used optimizer AdamW, set the number of iterations to 65, initialize the learning rate to 1×10^{-3}, and identify the local optimum using the cosine annealing algorithm. We optimize the linear encoder and attention encoder using a self-supervised mechanism to minimize the $\mathcal{L}_{res}$ and $\mathcal{L}_{cre}$ loss functions. After completing the pre-training phase, we integrate the initially trained deep linear encoder and attention encoder into a unified module to form our model. Finally, according to the hyper-parameter experiment, when the hyper-parameters $\mu = 0.2$ and $\gamma = 1.0$, the experimental results reached the optimal solution, respectively.

4.3 Experimental results and analysis

To enhance the credibility of the experimental results, we conducted experiments on both real and synthetic datasets. We selected various evaluation indices for local community detection to assess the experimental performance. Considering the sensitivity of the initial seed to local community detection, we selected different numbers of nodes as initial seeds based on the dataset's node count to minimize the seeds' influence on the experimental outcomes. In the actual dataset, when the number of nodes is less than 5000, 3000 nodes are randomly chosen as initial seeds. For node numbers falling within the range of 5000 to 10000, 6000 nodes are selected as initial seeds. When the number of nodes ranges from 10000 to 100000, 10000 nodes are randomly chosen as initial seeds. If the number of nodes exceeds 100000, 100000 random seeds are selected for local community detection experiments to assess the method's competitiveness. In addition, in the synthetic dataset, half of the total number of nodes is used as the initial seed number to enhance the interpretability of the experimental results. Finally, the ultimate experimental result is the average value of the same evaluation index from different nodes within a single dataset.

4.3.1 The results and analysis on real-world networks

This experiment evaluated the disparity between the acquired community structure and the actual ground community by analyzing various evaluation indicators across different datasets and baseline algorithms. F_1-score, Con, and local mutual information value NMI_{Local} of the resulting communities are presented in Tables 4, 5, and 6, respectively. It can be observed from the results in these tables that across different evaluation metrics, the LCDCN method outperforms nearly all other approaches. This superiority can primarily be attributed to several key inherent factors within LCDCN. Especially when compared with the latest deep-embedded work EL-Trans, it surpasses EL-Trans by more than twice on various indicators. This is because our work, on the one hand, better handles the relationship between neighboring nodes through novel attention weights, obtaining more comprehensive structural information. On the other hand, our model effectively introduces a supervised optimization mechanism to guide the model towards the desired direction. In addition, it is worth noting that the LCDCN performs well on relatively small-scale datasets such as Polblogs and Mich. It also achieves the highest scores on large-scale datasets like Slashdot and even on super-large-scale datasets like Amazon. There are two main reasons for this phenomenon. Firstly, our selection strategy can better select core seeds as the starting point of community detection, thereby improving the accuracy of community

Table 4 The scores of F_1 on real-world networks

Data	EL-Trans	SDCN	DFCN	ASFWF	LCDS-A	LCDPC	LCDMD	LCDCN
Polblogs	0.6246	0.6738	0.6604	0.7071	0.6955	0.7014	0.7428	**0.8343**
Mich	0.6482	0.6837	0.6128	0.7754	0.6978	0.7021	0.7478	**0.8133**
Dblp	0.4158	0.4619	0.4876	0.7152	0.5043	0.6977	0.7071	**0.7237**
Orkut	0.6207	0.6221	0.5989	**0.6876**	0.5541	0.5875	0.6804	0.6599
BlogCatalog	0.4549	0.5338	0.5892	0.6418	0.6327	0.5988	0.6095	**0.7031**
Wikipedia	0.6087	0.6603	**0.6620**	0.6975	0.5942	0.6119	0.6228	0.6559
HepPh	0.4768	0.4967	0.4837	0.5683	0.4203	0.4723	0.5015	**0.5849**
Slashdot	0.4463	0.4361	0.4088	**0.6629**	0.4978	0.5361	0.5796	0.6585
WebStanford	0.3098	0.4297	0.4496	0.5997	0.2332	0.4987	0.5048	**0.6831**
Amazon	0.2879	0.3212	0.3316	0.4768	0.4921	0.5079	0.5226	**0.6231**

Table 5 The scores of *Con* on real-world networks

Data	EL-Trans	SDCN	DFCN	ASFWF	LCDS-A	LCDPC	LCDMD	LCDCN
Polblogs	0.4020	0.4496	0.5768	0.2968	0.3208	0.3230	0.2921	**0.2033**
Mich	0.5889	0.4766	0.5055	0.4142	0.4933	0.4295	0.4683	**0.3991**
Dblp	0.5743	0.5428	0.4896	**0.3729**	0.4933	0.5585	0.3946	0.3985
Orkut	0.5452	0.5438	0.6877	0.4591	0.4977	0.5182	0.4858	**0.4065**
BlogCatalog	0.4923	0.5438	0.5234	0.3996	0.4870	0.4256	0.4098	**0.3999**
Wikipedia	0.4044	0.4496	0.3768	**0.3092**	0.316	0.4579	0.3424	0.3183
HepPh	0.5342	0.5455	0.4964	0.4668	0.5002	0.4886	0.4827	**0.4663**
Slashdot	0.5978	0.6687	0.6238	0.5053	0.5326	0.5895	**0.5097**	0.5123
WebStanford	0.6632	0.6993	0.6460	0.6011	0.5874	0.5927	0.4908	**0.4837**
Amazon	0.6786	0.5893	0.5533	0.5024	0.5437	0.5621	**0.4438**	0.4667

Table 6 The scores of NMI_{Local} on real-world networks

Data	EL-Trans	SDCN	DFCN	ASFWF	LCDS-A	LCDPC	LCDMD	LCDCN
Polblogs	0.4836	0.5748	0.6069	0.7955	0.7364	0.7558	0.6189	**0.8134**
Mich	0.5208	0.5030	0.4998	0.7425	0.6957	0.7011	0.6748	**0.7956**
Dblp	0.5206	0.4703	0.4923	**0.6435**	0.5725	0.6025	0.5947	0.6347
Orkut	0.4998	0.4837	0.4232	0.5923	0.5124	0.5026	**0.5937**	0.5930
BlogCatalog	0.4182	0.3997	0.4205	0.5223	0.5033	0.4978	0.5034	**0.5991**
Wikipedia	0.5931	0.5537	0.5025	0.6997	0.6341	0.4978	**0.7021**	0.6938
HepPh	0.4131	0.4852	0.4408	0.5984	0.5023	0.5850	0.5376	**0.6000**
Slashdot	0.3956	0.3846	0.4011	0.5323	0.5028	0.5527	0.5413	**0.5855**
WebStanford	0.2224	0.3703	0.3406	0.4447	0.4078	0.3585	0.4078	**0.4485**
Amazon	0.1628	0.2856	0.2497	0.5621	0.4432	0.4998	0.5108	**0.6233**

detection from the source. Secondly, setting the number of seeds chosen in small-scale and super-large-scale datasets can reduce the algorithm's time and memory overhead and make the local community discovery work independent of the graph's size. By doing so, LCDMD still demonstrates excellent performance on large-scale datasets. In contrast, the deep embedding method EL-Trans and the deep clustering method DFCN cannot avoid this limitation, which results in relatively poor experimental outcomes.

However, as can be seen from Tables 4, 5, and 6, our proposed method also has a few limitations, resulting in a slightly lower amount of data than the baseline method. This phenomenon occurs especially in the Wikipedia dataset because, despite our emphasis on the influence of neighboring nodes on the structure, in the case of Wikipedia, which has a high node density, nodes with more neighbors tend to prioritize neighbor-related information. Especially when the number of neighboring layers increases, the algorithm's complexity significantly rises, leading to a decrease in algorithm performance. Therefore, our proposed method will be more suitable for scenarios with a slightly lower node density in the graph.

4.3.2 The results and analysis on synthetic networks

To ensure experimental fairness, we adopted the same number of initial seed nodes as in the real network and calculated the evaluation index scores using identical methods. We evaluated the performance of our proposed method LCDCN against baseline approaches and four synthetic networks of varying sizes. Table 7 presents F_1, while Tables 8 and 9 display *Con* and NMI_{Local} on the four synthetic networks. The experimental results demonstrate that LCDCN consistently outperforms other methods across all datasets and evaluation indexes, indicating its ability to detect local community structures resembling those in real communities accurately. Notably, LCDCN achieves near-perfect alignment with actual communities on the small-scale LFR-1 dataset. This can be attributed to the advantages offered by synthetic networks for maximizing LCDCN's performance and LCDCN's capability to learn optimal node features and structural information from smaller datasets. Furthermore, on the large-scale LFR-4 dataset, LCDCN demonstrates outstanding performance, surpassing all baseline methods in every index, considered state-of-the-art techniques. One crucial factor contributing to this achievement is that baseline methods are constrained by sensitivity towards seed node selection and dataset size sensitivity; thus, they fail to outperform our proposed method LCDNN regarding algorithmic performance. From Table 9 depicting NMI_{Local} on the LFR-3 dataset, ASFWF demonstrates a slight advantage over other approaches. However, despite mitigating seed selection sensitivity through integrating two different strategies within the ASFWF framework, it fails to effectively integrate information from diverse types of nodes, resulting in suboptimal solutions at a local level compared to our approach - LCDCN. Therefore, we can conclude that LCDCN can stably achieve the best results for different datasets.

4.4 Hyper-parameters analysis

In our experiments, the values of three hyper-parameters μ, γ, and k directly determine the performance of our model. The first hyper-parameter, μ, functions to combine attribute information and structural information with varying weights. The second hyper-parameter is γ, which controls the proportion of clustering loss. The last hyper-parameter is k, which represents the local order. Therefore, we conducted various experiments to assess the influence of different hyper-parameters on the outcomes and to employ suitable models to achieve the optimal solution.

4.4.1 Analysis of hyper-parameter μ

As previously stated, μ is the control parameter that integrates attribute information into structural information. Only reasonable fusion can make the combined information more accurate. To achieve the best solution, we uniformly select eleven points in the range [0,1] and calculate F_1 and NMI_{Local} using various data sets. We aim to identify the most

Table 7 The scores of F_1 on synthetic networks

Data	EL-Trans	SDCN	DFCN	ASFWF	LCDS-A	LCDPC	LCDMD	LCDCN
LFR-1	0.7279	0.7388	0.7221	0.9690	0.9005	0.8832	0.9206	**1.0000**
LFR-2	0.6747	0.6822	0.6806	0.9281	0.7832	0.7307	0.8733	**0.9529**
LFR-3	0.5654	0.5250	0.5995	0.8871	0.7058	0.7990	0.7956	**0.8982**
LFR-4	0.5211	0.5119	0.5376	0.7511	0.7118	0.6902	0.7845	**0.8581**

Table 8 The scores of *Con* on synthetic networks

Data	EL-Trans	SDCN	DFCN	ASFWF	LCDS-A	LCDPC	LCDMD	LCDCN
LFR-1	0.3835	0.3366	0.3276	0.2017	0.3149	0.3341	0.2920	**0.1778**
LFR-2	0.3567	0.3088	0.3254	0.2472	0.3971	0.3072	0.3300	**0.1996**
LFR-3	0.4455	0.4647	0.4896	0.4063	0.5019	0.4972	0.3998	**0.2019**
LFR-4	0.6448	0.6606	0.5996	0.4895	0.6482	0.6219	0.5210	**0.3098**

Table 9 The scores of NMI_{Local} on synthetic networks

Data	EL-Trans	SDCN	DFCN	ASFWF	LCDS-A	LCDPC	LCDMD	LCDCN
LFR-1	0.7089	0.7356	0.7402	0.8447	0.7028	0.6988	0.8021	**0.9129**
LFR-2	0.6674	0.6922	0.6218	0.7548	0.6509	0.6532	0.7092	**0.8859**
LFR-3	0.6061	0.5948	0.5768	**0.8189**	0.7010	0.5956	0.6650	0.8093
LFR-4	0.4950	0.4837	0.4578	0.6927	0.5867	0.6654	0.6982	**0.7012**

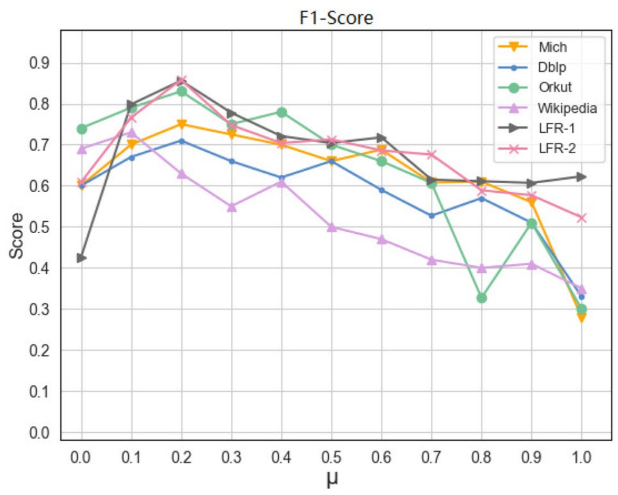


Fig. 2 The score of F_1 with hyper-parameter μ in different datasets

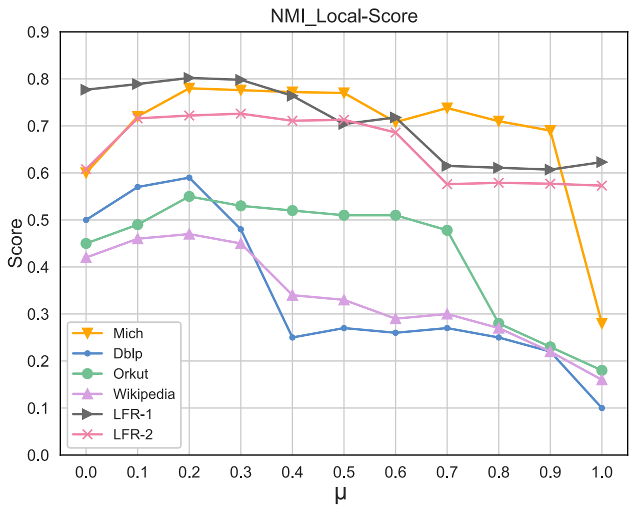


Fig. 3 The score of NMI_{Local} with hyper-parameter μ in different datasets

appropriate value through multiple iterations and verifications. The results are shown in Figs. 2 and 3.

It is not difficult to see from Figs. 2 and 3 that when $\mu = 0.2$, the model's performance reaches its peak. As the value of μ increases gradually, the final score of the model decreases slowly. The reason for this result is that attribute information complements structural information. Therefore, incorporating attribute information will undoubtedly enhance the model's performance. It is important to note that structural information in a graph contains significant data that can more directly reflect the graph's structural characteristics than attribute information. Therefore, the attribute information of the graph complements and enhances the structural information.

4.4.2 Analysis of hyper-parameter γ

The second hyper-parameter γ appears in Eq. (24). As shown in Eq. (24), its role is to balance the optimization term in the loss function, which will control the performance of community discovery. We deliberately selected several odd numbers of varying magnitudes to investigate the influence of various values γ on the outcomes. The purpose of selecting odd numbers is that it allows for arbitrary distribution of γ. It can be punitive to discourage the community from moving in the wrong direction. In the same way, it can also serve as a reward item to incentivize community detection more effectively. Therefore, we select representative numbers at different intervals to validate the experimental results and to determine the most appropriate values to enhance experimental performance. The F_1 and NMI_{Local} in different datasets shown in Fig. 4a, b.

As shown in the four subfigures of Fig. 4, when $\gamma = 1$, the model's performance reaches its optimal level. In addition, it is not difficult to see from the figure that although the range of parameters is extensive, the impact on the experimental results is minimal. This is because our method is robust, which ensures stable results. Consequently, the optimization parameters remain stable, enhancing its optimization performance.

4.4.3 Analysis of hyper-parameter k

k is a hyper-parameter in the k-subgraph extractor. Specifically, k determines the number of layers to traverse the neighbors around the seed node. On the one hand, having too many layers can cause the experiment's memory consumption exponentially, eventually leading to memory overflow, which is not conducive to the experiment. On the other hand, having too many levels will increase the experiment's complexity, leading to time consumption challenges. On the contrary, having too few levels may result in incomplete local community detection and an inadequate graph structure. Therefore, selecting an appropriate value for k ensures experimental accuracy and complexity. Figure 5a, b illustrate the influence of the selection of k in different datasets on F_1 and NMI_{Local}.

When k is around 3, we observed that the models exhibit the best performance. When k does not exceed 3, the impact of the k-subgraph extractor on the experiment is relatively large. This is because too little subgraph information cannot enhance the performance of local community detection. On the contrary, it will reduce the performance of community detection. When k exceeds 3, the experimental performance gradually stabilizes, but there is also a slight decreasing trend. The reason for this phenomenon is that as the number of layers increases, the complexity of the experiment also increases, leading to memory overflow, especially in datasets

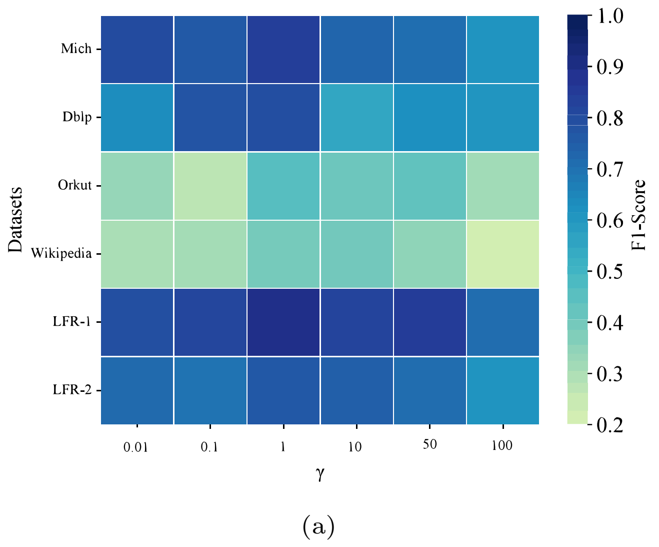


Fig. 4 The score of F_1 and NMI_{Local} with hyper-parameter γ in different datasets

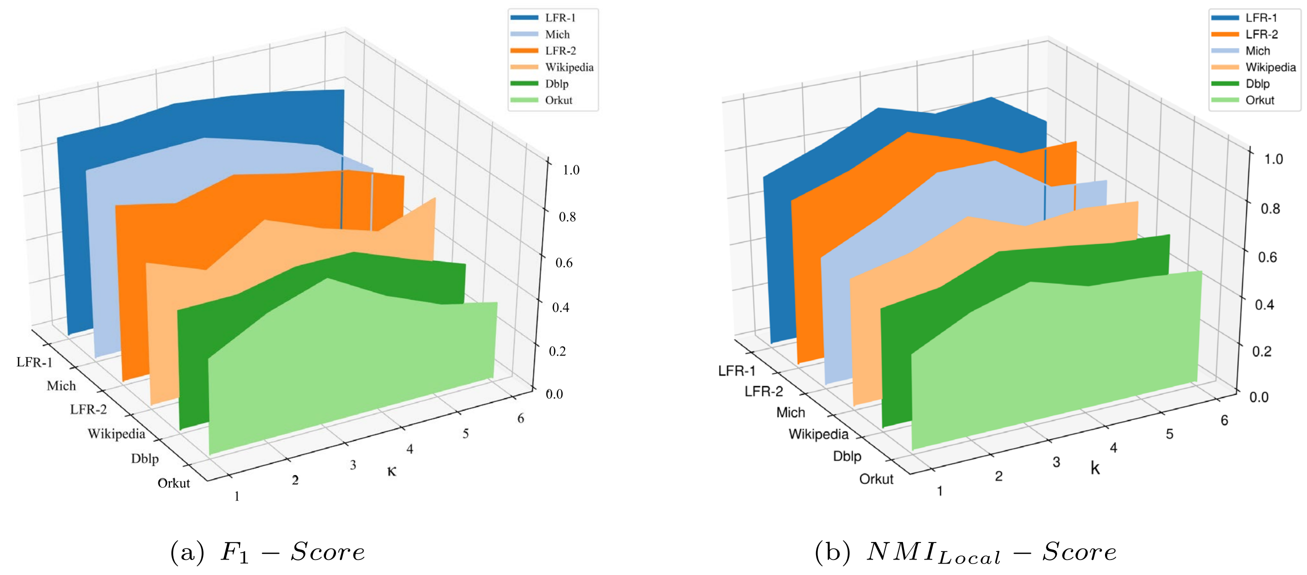
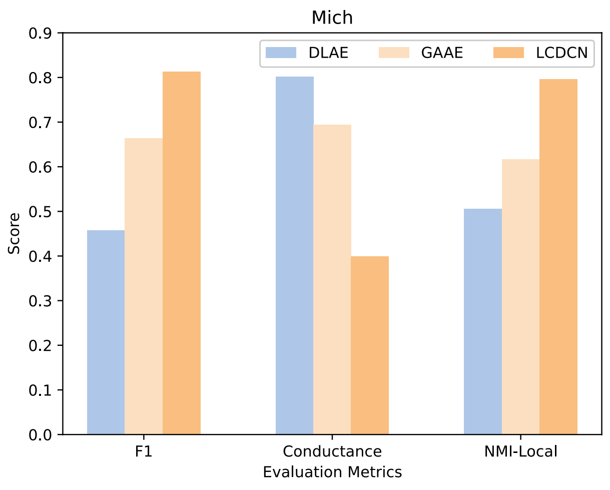


Fig. 5 The score of F_1 and NMI_{Local} with hyper-parameter k in different datasets

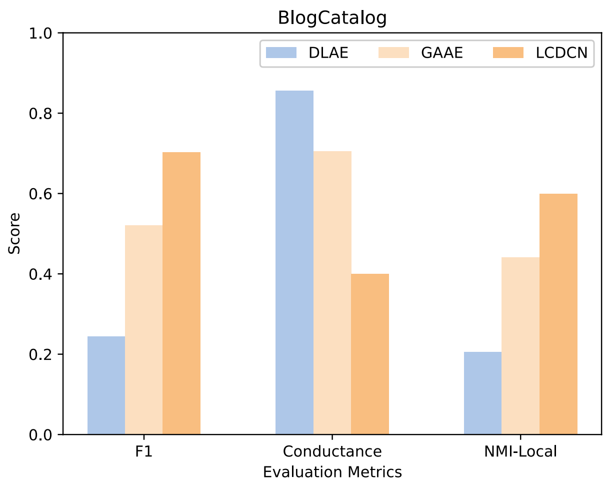
with a large average node degree. This phenomenon is more pronounced, ultimately resulting in poor experimental performance. It cannot continue to increase or even decrease.

4.5 Ablation studies

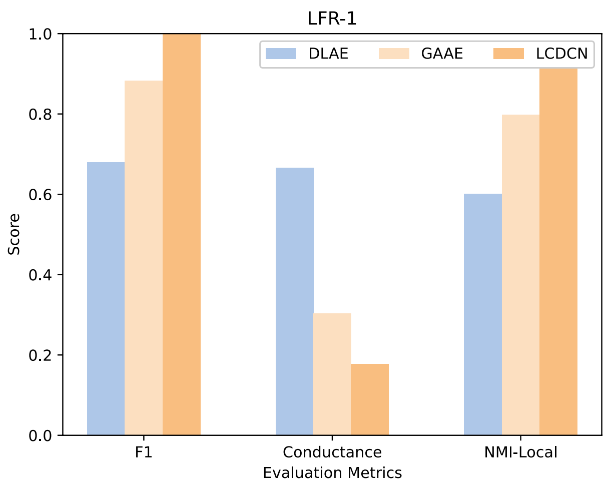
In this section, we conduct ablation experiments to validate the efficacy of different modules in LCDCN, and the results are presented in Figs. 6, 7, and 8. Specifically, first of all, we employ the DLAE, GAAE, as well as their synthesis LCDCN to generate feature representations, respectively. These methods were used to compute the corresponding evaluation index results on two real and two synthetic datasets, and the findings are presented in Fig. 6a,b,c,d. Figure 6 reveals that using DLAE or GAAE alone fails to achieve comparable performance to LCDCN; in fact, it falls significantly short even by less than half of its results. This discrepancy can be attributed to DLAE's inability to accurately capture complex graph data structures solely based on attribute information, which inadequately reflects real structural states. Similarly, although GAAE exhibits improved performance compared to DLAE in capturing structural information for community detection tasks, it still falls short compared to LCDCN due to its failure to capture node



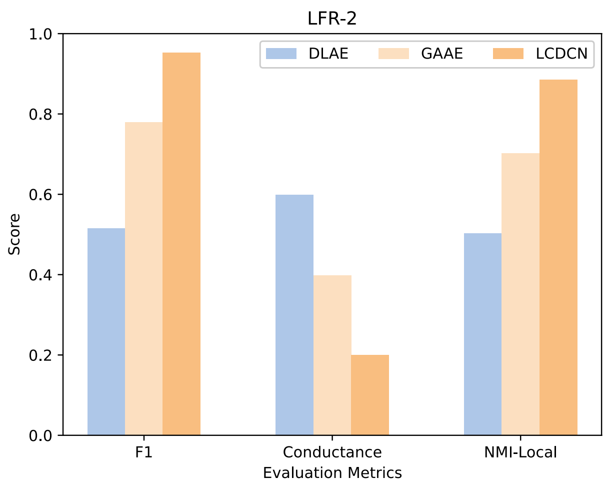
(a)



(b)



(c)



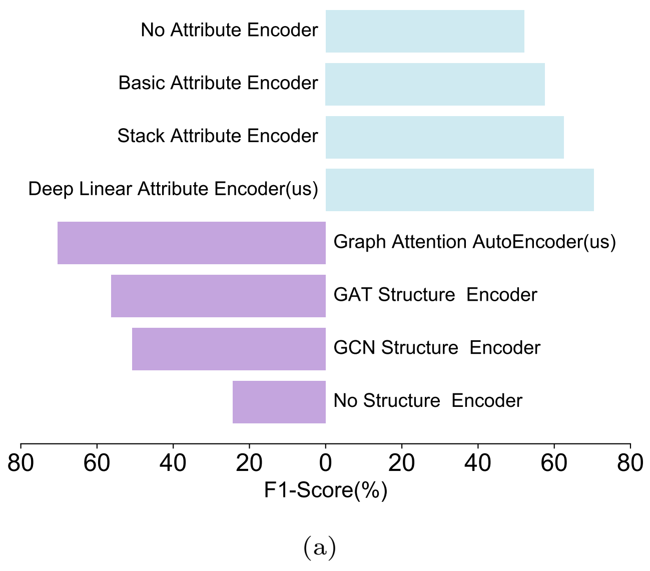
(d)

Fig. 6 Local communities detected results by F_1-Score, *Con*, and NMI_{Local} based on different datasets

attribute information, which is crucial for a comprehensive analysis of graph data. Therefore, from an individual module perspective, only through their organic integration can they mutually maximize potential and enhance the accuracy of community detection.

Subsequently, we proceeded with the design of ablation experiments by individually substituting the DLAE module for capturing attribute information and the GAAE module for capturing structure information. Additional experimental results are presented in Figs. 7 and 8. Specifically, we initially replaced the DLAE module with an ordinary, simple, basic attribute encoder and a stacked attribute encoder. Furthermore, our approach substituted graph convolutional network (GCN) and graph attention network (GAT) for the GAAE module. Finally, we experimented with the replaced components on the real dataset BlogCatalog and the synthetic dataset LFR-2. We evaluated the scores of different method combinations on the two evaluation metrics F_1 and NMI_{Local}.

The results presented in Figs. 7 and 8 demonstrate that the various modules of LCDCN consistently outperform other methods of the same type, both on real and synthetic datasets. This indicates that our model's module components perform better than other methods with similar functionality. Several reasons contribute to this observation: Firstly, the deep linear encoder in LCDCN significantly enhances learning generalization ability by incorporating a unique bias term. Secondly, unlike traditional approaches, the attention encoder in our model reconstructs attention coefficients based on neighboring nodes, enabling the fusion



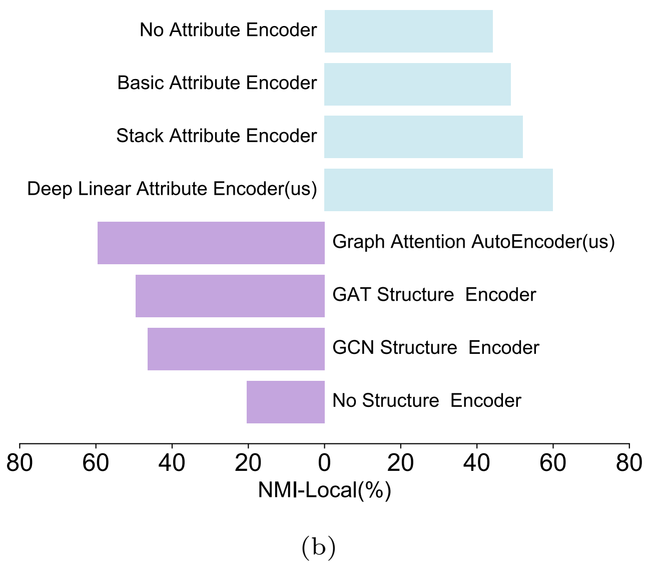
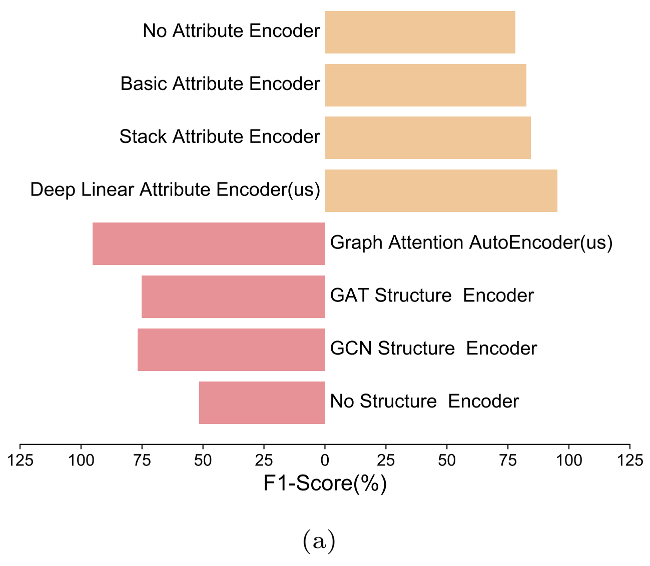


Fig. 7 Local community detection results of F_1-Score and NMI_{Local} based on attribute and structural Encode on BlogCatalog dataset



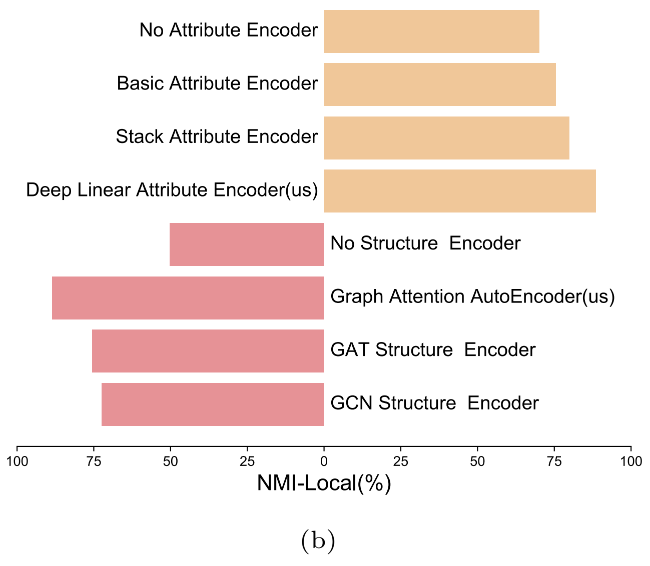


Fig. 8 Local community detection results of F_1-Score and NMI_{Local} based on different attribute and structural encode on LFR-2 dataset

of higher-order information as an attention mechanism within LCDCN. Consequently, its performance surpasses graph neural networks and traditional attention networks. Additionally, we observed a mild and insignificant decline in experimental performance when excluding the attribute encoding component; however, omitting the structure encoding component resulted in a significant overall decrease in performance. This finding confirms that attribute information is complementary while structural information dominates graph data structures. In summary, each module of LCDCN exhibits strong capabilities for processing node information.

In addition, it is evident from Figs. 7 and 8 that there exists a significant distinction between encoders lacking attribute encoding and structure encoding and those with encoders. This proves the importance of both attribute and structural information for graph data structures. Furthermore, it is observed from Figs. 7 and 8 that the basic and stack attribute encoders have minimal impact on the experimental results, indicating that more than simply stacking encoders does not notably enhance model performance. In contrast, DLAE effectively integrates multiple layers of information through newly added bias to improve performance. Similarly, the influence of GAT and GCN structure encoders on experimental results is inconspicuous due to their similar framework to GAT's reliance on the GCN framework. While GAT proposes the attention coefficients fusion neighbor information, it fails to consider the substructure around the node. In contrast, GAAE considers the local substructure around each node and fuses it with structural attention. This enhancement results in more abundant structure information.

4.6 Time complexity analysis

In this paper, we assume that the number of nodes is N, the number of edges is $|E|$, and the output dimension of each layer of the encoder is $d_1, d_2, ..., d_L$. Since DLAE focuses on node features, its time complexity is $O(Nd_1d_2...d_L)$. For GAAE, its time complexity consists of two parts: feature mapping and attention calculation. Therefore, its complexity is approximately $O(Nd_1d_2...d_L) + |E|d_1d_2...d_L)$ $=O((N + |E|)d_1d_2...d_L)$. Finally, the complexity of the self-supervised optimization of formula 20 is $O(NlogN)$. So, the total time complexity is $O((N + |E|)d_1d_2...d_L) + O(NlogN)$.

To verify the time complexity, we compare the time with three deep learning baseline methods and a traditional local community detection method. Our goal is to find communities similar to the seed node. Our timing starts from the beginning of the algorithm and ends when the goal is achieved. We choose two datasets of different sizes. For each dataset, we calculate the results at various scales. Figure 9 shows the relationship between the running time of the method and the network size.

The experimental results show that our method has sufficient advantages. The main reasons for this are as follows. First, existing deep learning methods focus on the global graph information, which increases the time to visit irrelevant nodes for local community detection. Such operations will lead to an increase in time complexity. Secondly, as the number of nodes increases, traditional methods gradually lose competitiveness.

In addition, we let the input of baseline methods SDCN and DFCN no longer be the global graph information but a local k-subgraph. The experimental results are shown in Fig. 10. We found that the complexity of our method cannot reach the optimal level but is infinitely close to them. Our process focuses on the local structural information of the node, resulting in a slightly higher complexity than the method that only focuses on the node itself. However, the results of the experimental accuracy show that such a sacrifice is worthwhile.

4.7 Visualization analysis

To demonstrate our algorithm LCDCN's superiority, we visually demonstrated the experimental results and the baseline method DFCN and ASFWF to show the differences between LCDCN and the baseline method. Specifically, we used the same random seed node for local community detection on the small dataset LFR-0 and compared it with the DFCN and ASFWF algorithms. Figure 11 shows the experimental results. In particular, Fig. 11a shows the real community structure of LFR-0,

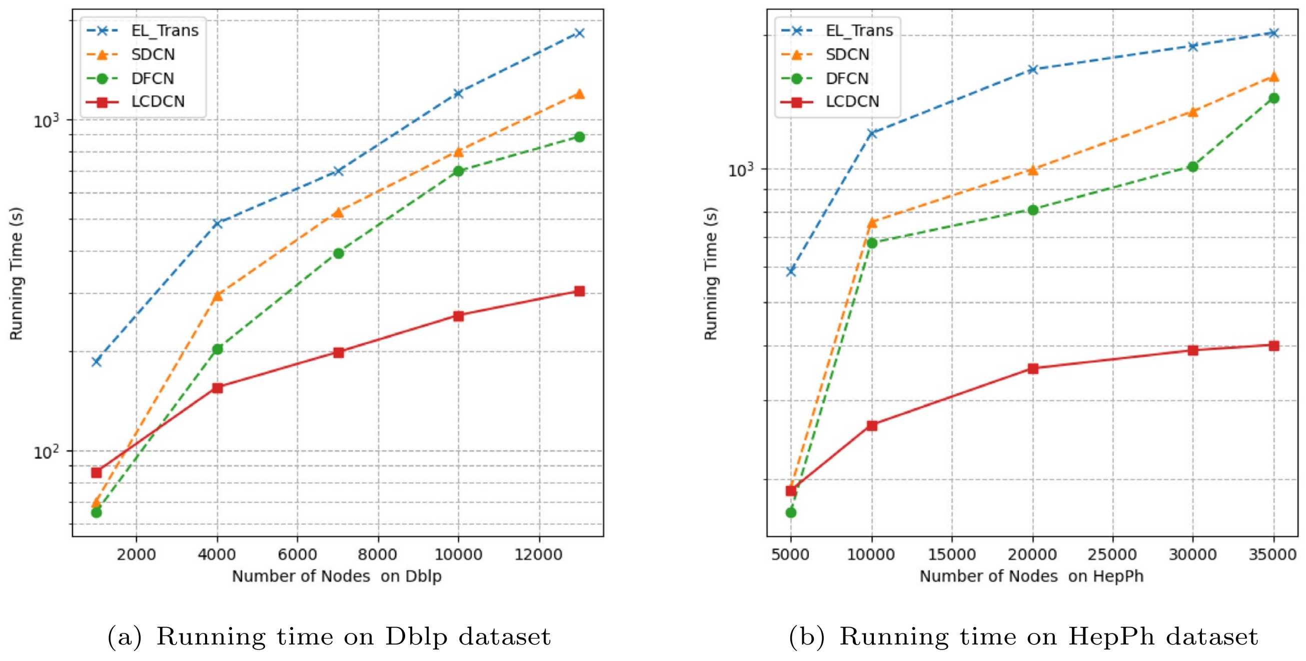
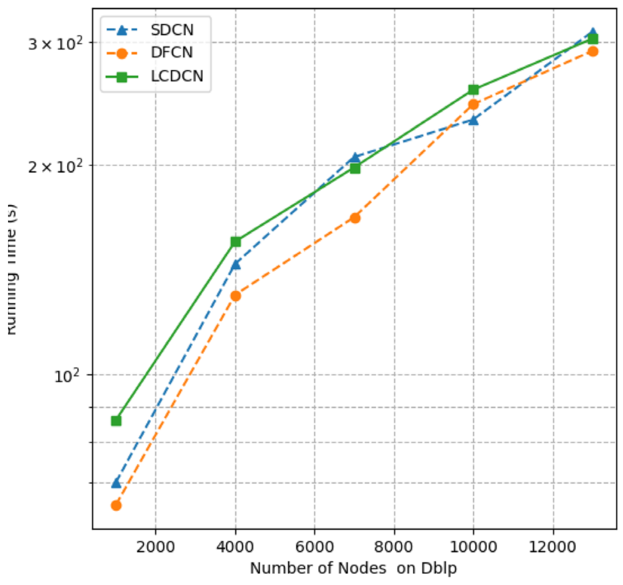
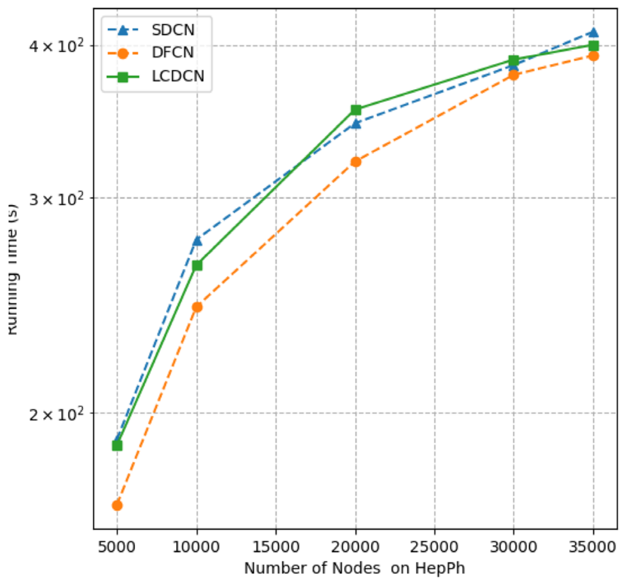


Fig. 9 Algorithm runtime comparison on global graph



(a) Running time on Dblp dataset



(b) Running time on HepPh dataset

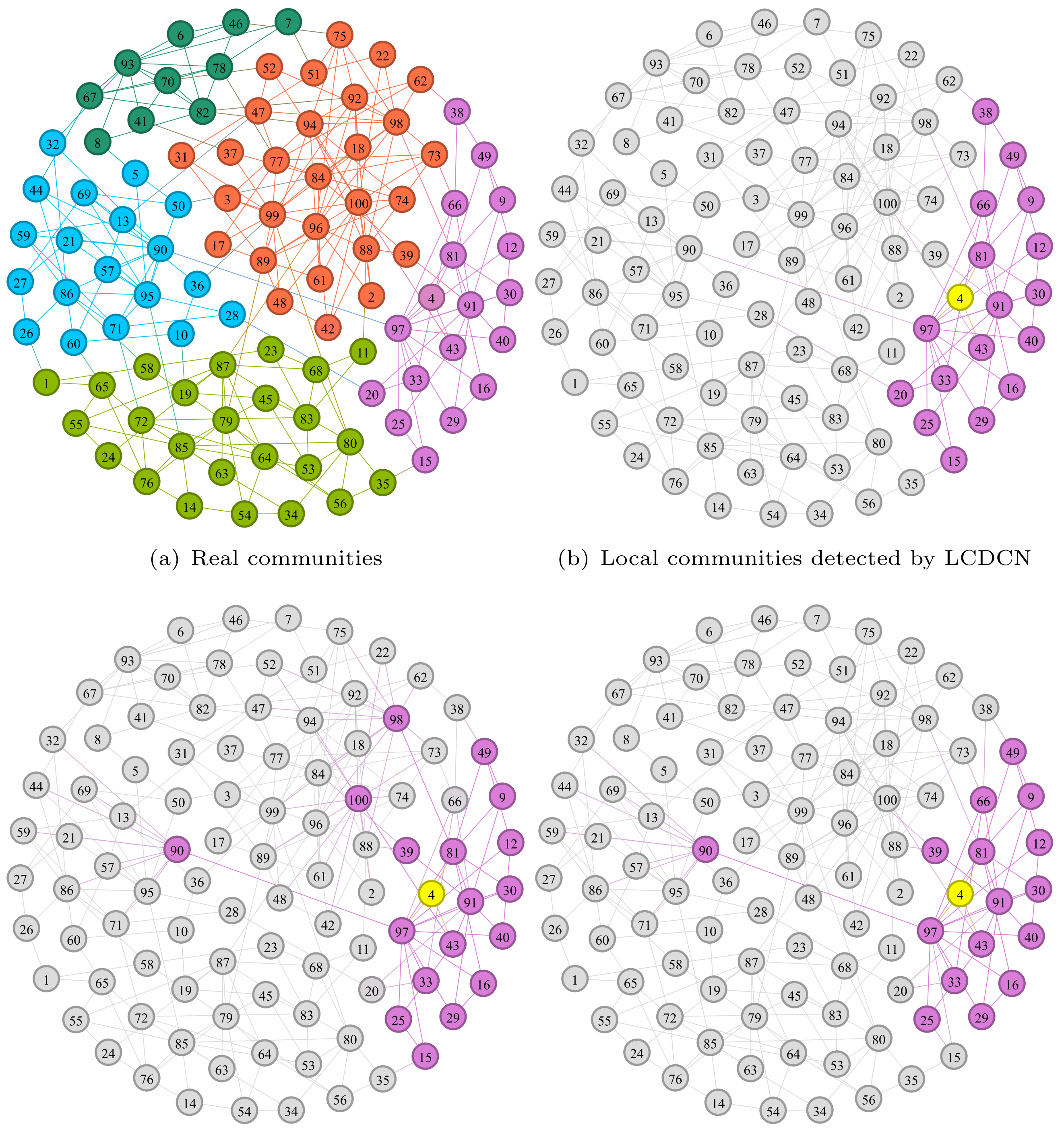
Fig. 10 Algorithm runtime comparison on *k*-subgraph

and a total of 5 communities can be observed. Figure 11b shows the visual demonstration results of our method LCDCN. Figures 11c, d show the baseline method's local community visualization results. To avoid the occasional high-accuracy error caused by selecting a special seed, we decided to use edge node 4 as the initial seed node. The set of purple nodes represents the detected local community structure.

It can be observed from the four subgraphs in Fig. 11 that our algorithm can accurately detect communities consistent with real tags. The LCDCN algorithm itself has high accuracy, even in the case of edge node 4. The main reason is that we find the most recent and most important node in the local area through any node to divide the community. This largely avoids the situation of the real community being unable to be found through edge nodes. However, some nodes are misidentified and incorrectly identified in the baseline methods DFCN Fig. 11c and ASFWF Fig. 11d. After observation, it is found that the main reason for this situation is that the baseline method is not sensitive to edge nodes and even regards other community top core nodes 90 as the same label as node 4. In addition, this is closely related to the characteristics of the original algorithm itself, which makes the algorithm only focus on the local state of seed nodes and ignore the community edge nodes, resulting in a decrease in ASFW and DFCN accuracy. Because of the above reasons, LCDCN is ahead of the most advanced baseline methods.

5 Conclusions

This paper comprehensively discusses local community detection work, addressing current challenges and proposing an LCDCN framework for local community detection. Firstly, LCDCN addresses the critical issue of selection in local community detection by considering the high-order neighbor structure and utilizing similarity and correlation as influencing factors for selection. This approach ensures that the selected seed solution aligns more consistently with core node characteristics. Secondly, the reconstructed *k*-subgraph provides convenience for large-scale networks and can be expanded and applied in local community detection work on large graphs. Lastly, a new node encoding method is employed to comprehensively obtain structural information and attribute information for clustering purposes, significantly improving the accuracy of local community detection tasks. Experimental results also confirm the effectiveness of LCDCN in local community detection tasks. In future work, we will continue to explore multi-view approaches to local community detection and address interference caused by seed nodes belonging to overlapping communities. By employing multi-view techniques, we aim to identify each different community where the seed node is located instead of a single or merged community. Additionally, we will enhance algorithm robustness and universality by introducing dynamic algorithm strategies.



(a) Real communities

(b) Local communities detected by LCDCN

(c) Local communities detected by DFCN

(d) Local communities detected by ASFWF

Fig. 11 Real communities and local communities detected by LCDCN, DFCN, and ASFWF on node 4 of the LFR-0 network

Author Contributions XG and XL: conceptualization, methodology, formal analysis, software, investigation, validation, resources, writing-original draft, review and editing, visualization. WS and SW: resources, writing-review and editing, supervision.

Funding No funding was received to assist with the preparation of this manuscript.

Data Availability The datasets analyzed during the current study were all derived from the following public domain resources.

Declarations

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

Consent to participate The authors declare that they agree to participate.

Consent for publication The authors declare that they agree to publish.

References

1. Li J, Lai S, Shuai Z, Tan Y, Jia Y, Yu M, Song Z, Peng X, Xu Z, Ni Y et al (2024) A comprehensive review of community detection in graphs. Neurocomputing 600:128169
2. Li X, Peng Q, Li R, Ma H (2024) Dual graph neural network for overlapping community detection. J Supercomput 80(2):2196–2222
3. Chen Y, Du L, Zhou P, Duan L, Qian Y (2024) Multiple kernel clustering with local kernel reconstruction and global heat diffusion. Inform Fus 105:102219
4. Wasserman S, Faust K (1994) Social Network Analysis: Methods and Applications. Cambridge University Press, Cambridge
5. Ito T, Chiba T, Ozawa R, Yoshida M, Hattori M, Sakaki Y (2001) A comprehensive two-hybrid analysis to explore the yeast protein interactome. Proc Natl Acad Sci 98(8):4569–4574
6. Luo X, Liu Z, Shang M, Lou J, Zhou M (2020) Highly-accurate community detection via pointwise mutual information-incorporated symmetric non-negative matrix factorization. IEEE Trans Netw Sci Eng 8(1):463–476
7. Ni L, Ge J, Zhang Y, Luo W, Sheng VS (2023) Semi-supervised local community detection. IEEE Trans Knowl Data Eng 36(2):823–839
8. Baltsou G, Christopoulos K, Tsichlas K (2022) Local community detection: a survey. IEEE Access 10:110701–110726
9. Chen G, Zhou S (2024) A novel overlapping community detection strategy based on core-bridge seeds. Int J Mach Learn Cybernet 15(6):2131–2147
10. Samie ME, Behbood E, Hamzeh A (2023) Local community detection based on influence maximization in dynamic networks. Appl Intell 53(15):18294–18318
11. Lee B, Su Y, Kong Q, Zhang T (2024) Enhanced multi-view anomaly detection on attribute networks by truncated singular value decomposition. International Journal of Machine Learning and Cybernetics, 1–19
12. Williams F, Huang J, Swartz J, Klar G, Thakkar V, Cong M, Ren X, Li R, Fuji-Tsang C, Fidler S et al (2024) fvdb: A deep-learning framework for sparse, large scale, and high performance spatial intelligence. ACM Transactions on Graphics (TOG) 43(4):1–15
13. Tu W, Zhou S, Liu X, Guo X, Cai Z, Zhu E, Cheng J (2021) Deep fusion clustering network. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp 9978–9987
14. Zhang W, Yang M, Sheng Z, Li Y, Ouyang W, Tao Y, Yang Z, Cui B (2021) Node dependent local smoothing for scalable graph learning. Adv Neural Inform Process Syst 34:20321–20332
15. Girvan M, Newman ME (2002) Community structure in social and biological networks. Proc Natl Acad Sci 99(12):7821–7826
16. Newman ME (2006) Modularity and community structure in networks. Proc Natl Acad Sci 103(23):8577–8582
17. Zhu W, Sun Y, Fang R, Xu B (2023) A low-memory community detection algorithm with hybrid sparse structure and structural information for large-scale networks. IEEE Trans Parallel Distributed Syst 34(10):2671–2683
18. Zeng G, Peng H, Li A, Wu J, Liu C, Philip SY (2025) Scalable semi-supervised clustering via structural entropy with different constraints. IEEE Trans Knowl Data Eng 37(1):478–492
19. Peng H, Zhang J, Huang X, Hao Z, Li A, Yu Z, Yu PS (2024) Unsupervised social bot detection via structural information theory. ACM Trans Inform Syst 42(6):1–42
20. Sheykhzadeh J, Zarei B, Gharehchopogh FS (2024) Community detection in social networks using a local approach based on node ranking. IEEE Access 12(6):92892–92905
21. Liu J, Shao Y, Su S (2021) Multiple local community detection via high-quality seed identification over both static and dynamic networks. Data Sci Eng 6(3):249–264
22. Yin H, Benson AR, Leskovec J, Gleich DF (2017) Local higher-order graph clustering. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 555–564
23. Clauset A (2005) Finding local community structure in networks. Phys Rev E 72(2):026132
24. Shang R, Zhang W, Jiao L, Stolkin R, Xue Y (2017) A community integration strategy based on an improved modularity density increment for large-scale networks. Physica A 469:471–485
25. Zhang W, Zhao K, Shang R (2024) Evolutionary multi-objective attribute community detection based on similarity fusion strategy with central nodes. Appl Soft Comput 150:111101
26. Luo W, Lu N, Ni L, Zhu W, Ding W (2020) Local community detection by the nearest nodes with greater centrality. Inform Sci 517:377–392
27. Shang R, Zhang W, Zhang J, Jiao L, Li Y, Stolkin R (2022) Local community detection algorithm based on alternating strategy of strong fusion and weak fusion. IEEE Trans Cybernet 53(2):818–831
28. 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
29. Wang C, Pan S, Hu R, Long G, Jiang J, Zhang C (2019) Attributed graph clustering: A deep attentional embedding approach. In: International Joint Conference on Artificial Intelligence 2019,(IJCAI), pp. 3670–3676
30. Li D, Zhong X, Dou Z, Gong M, Ma X (2021) Detecting dynamic community by fusing network embedding and nonnegative matrix factorization. Knowl-Based Syst 221:106961
31. Zhou X, Su L, Li X, Zhao Z, Li C (2023) Community detection based on unsupervised attributed network embedding. Expert Syst Appl 213:118937
32. Li D, Ma X, Gong M (2021) Joint learning of feature extraction and clustering for large-scale temporal networks. IEEE Trans Cybernet 53(3):1653–1666
33. Wu X, Lu W, Quan Y, Miao Q, Sun PG (2024) Deep dual graph attention auto-encoder for community detection. Expert Syst Appl 238:122182
34. Sun K, Qiu L, Zhao W (2024) Aegraph: Node attribute-enhanced graph encoder method. Expert Syst Appl 236:121382
35. Liu Z, Zuo W, Zhang D, Zhou C (2024) Self-attention enhanced auto-encoder for link weight prediction with graph compression. IEEE Trans Netw Sci Eng 11(1):89–99
36. Hu B, Zang Z, Xia J, Wu L, Tan C, Li SZ (2023) Deep manifold graph auto-encoder for attributed graph embedding. In: ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp 1–5
37. Mokhtari M, Kherad M (2024) Community detection by influential nodes based on random walk distance. Concurrency Comput 36(17):8135–8156
38. Li Q, Ma H, Li J, Li Z, Chang L (2023) Attributed multi-query community search via random walk similarity. Inform Sci 631:91–107

39. Pan S, Wu J, Zhu X, Zhang C, Wang Y (2016) Tri-party deep network representation. In: International Joint Conference on Artificial Intelligence 2016, pp 1895–1901. Association for the Advancement of Artificial Intelligence (AAAI)
40. Hou C, He S, Tang K (2020) Rosane: Robust and scalable attributed network embedding for sparse networks. Neurocomputing 409:231–243
41. Xu K, Hu W, Leskovec J, Jegelka S (2019) How powerful are graph neural networks? In: International Conference on Learning Representations, pp 1–17
42. Wijesinghe A, Wang Q (2022) A new perspective on how graph neural networks go beyond weisfeiler-lehman? In: International Conference on Learning Representations, pp 1–23
43. Zhou X, Su L, Li X, Zhao Z, Li C (2023) Community detection based on unsupervised attributed network embedding. Expert Syst Appl 213:118937
44. Rossi RA, Ahmed NK (2015) The network data repository with interactive graph analytics and visualization. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp 4292–4293
45. Zhu J, Wang C, Gao C, Zhang F, Wang Z, Li X (2021) Community detection in graph: an embedding method. IEEE Trans Netw Sci Eng 9(2):689–702
46. Cavallari S, Zheng VW, Cai H, Chang KC-C, Cambria E (2017) Learning community embedding with community detection and node embedding on graphs. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp 377–386
47. Luo X, Liu Z, Shang M, Lou J, Zhou M (2020) Highly-accurate community detection via pointwise mutual information-incorporated symmetric non-negative matrix factorization. IEEE Trans Netw Sci Eng 8(1):463–476
48. Zhang Y, Wu B, Liu Y, Lv J (2019) Local community detection based on network motifs. Tsinghua Sci Technol 24(6):716–727
49. Yang J, Leskovec J (2012) Defining and evaluating network communities based on ground-truth. In: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics, pp 1–8
50. Lancichinetti A, Fortunato S, Radicchi F (2008) Benchmark graphs for testing community detection algorithms. Phys Rev E-Stat Nonlinear Soft Matter Phys 78(4):046110
51. Lin L, Li R, Jia T (2023) Scalable and effective conductance-based graph clustering. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, pp 4471–4478
52. Yang X-H, Ma G-F, Jin X, Long H-X, Xiao J, Ye L (2023) Knowledge graph embedding and completion based on entity community and local importance. Appl Intell 53(19):22132–22142
53. Bo D, Wang X, Shi C, Zhu M, Lu E, Cui P (2020) Structural deep clustering network. In: Proceedings of the Web Conference 2020, pp 1400–1410
54. Christopoulos K, Baltsou G, Tsichlas K (2023) Local community detection in graph streams with anchors. Information 14(6):332
55. Wang S, Yang J, Ding X, Zhang J, Zhao M (2023) A local community detection algorithm based on potential community exploration. Front Phys 11:1114296–1114312
56. Guo K, Huang X, Wu L, Chen Y (2022) Local community detection algorithm based on local modularity density. Appl Intell 52(2):1238–1253