Expert Systems With Applications 213 (2023) 118937

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Community detection based on unsupervised attributed network embedding

Xinchuang Zhou [a], Lingtao Su [a], Xiangju Li [a], Zhongying Zhao [a,*], Chao Li [a,b,*]

[a] School of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[b] School of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao, 266590, China

ARTICLE INFO

Keywords:
Community detection
Graph auto-encoder
Unsupervised representation learning

ABSTRACT

Community detection methods based on attribute network representation learning are receiving increasing attention. However, few existing works are focused exclusively on unsupervised network representation learning for the task of community detection. They mainly capture information about the topology or attributes of the network, but do not fully utilize clustering-oriented information. In this paper, we present a community detection algorithm based on unsupervised attributed network embedding (CDBNE) to resolve the above issues. To be specific, we propose a framework that learns the representation based on network structure and attribute information and the clustering-oriented representation simultaneously. The framework includes the graph attention auto-encoder module, the modularity maximization module, and the self-training clustering module. Firstly, CDBNE encodes the topology structure and the node attribute with the graph attention mechanism. Secondly, it captures the mesoscopic community structure with modularity maximization. Finally, the self-training clustering module optimizes the representation learning process in a self-supervised manner to obtain high-quality node representation. The performance of CDBNE is verified with experiments on community detection tasks. According to the results on three datasets, CDBNE outperforms the state-of-the-art methods. The implementation of CDBNE is available at https://github.com/xidizxc/CDBNE.

1. Introduction

Community detection can help uncover potential community structures in the network, which plays vital roles in various research domains (Gao et al., 2021). Thus community detection is widely used in various networks. For example, in social networks, community detection can help platform service providers find the groups with similar interests, which is beneficial for recommending items to targeted users (Mishra et al., 2021). In citation networks, community detection can identify new research teams and predict research trends (Nadimi-Shahraki et al., 2021). In biological networks, community detection can assist in revealing the community structure and evolution of certain biomolecules (Doluca & Oğuz, 2021).

However, traditional community detection methods suffer from problems of high computational complexity and huge storage space costs, which makes them hard to expand into large-scale networks with high-dimensional attributes (Naik et al., 2022). Network representation learning aims to transform nodes in topological space into low-dimensional vector space (Sun & Zhang, 2019), while maximally maintaining the topological structure and node attribute of the network. Therefore, community detection methods based on network representation learning has become a popular topic (Liu et al., 2020).

Early network representation learning methods learn the representation by Laplace feature mapping (Shao-hai et al., 2012) and random walk (Sun & Zhang, 2019). As deep learning becomes greatly popular, some researchers begin to study and design deep graph learning methods. For example, DNGR (Cao et al., 2016) uses network structure to construct the probabilistic co-occurrence matrix by employing the random surfing model. It designs a stacked denoising auto-encoder for learning the representations of graph nodes. ARGA and ARVGA (Pan et al., 2019) regularize a self-encoder by using the adversarial network, which forces the distribution of latent variables to match a Gaussian prior. The above self-encoder-based network representation learning methods mainly consider the topology structure of the network during the decoding process.

To further extract the attribute information of the network, some efforts are made in recent years (Xu et al., 2022). MGAE (Wang et al., 2017) follows the idea of denoising self-encoder to reconstruct randomly corrupted node attributes. It combines the marginalization process with spectral convolution on networks. GATE (Salehi & Davulcu, 2020) uses a self-encoder based on an attention mechanism to reconstruct the topology structure as well as the node attribute to obtain the final representation. SENet (Zhang et al., 2021) integrates topology

* Correspondence to: 579 Qianwangang Road, Economic & Technical Development Zone, Qingdao, Shandong Province, 266510 P.R. China.
E-mail addresses: zxc28432@163.com (X. Zhou), skd996721@sdust.edu.cn (L. Su), skd996730@sdust.edu.cn (X. Li), zyzhao@sdust.edu.cn, zzysuin@163.com (Z. Zhao), lichao@sdust.edu.cn (C. Li).

https://doi.org/10.1016/j.eswa.2022.118937
Received 29 July 2022; Received in revised form 10 September 2022; Accepted 27 September 2022
Available online 3 October 2022

structure and node attribute into the matrix of the kernel function through a high-order graph convolutional network. It uses spectral clustering loss to learn the node representation. The main idea of the above methods is to apply clustering methods based on the previously learned node representations (Jin et al., 2021). Wang et al. presented a clustering-oriented algorithm, which learns the weights and the node representation by an attention mechanism (Wang et al., 2019). It optimizes the reconstruction loss jointly with the clustering loss to obtain the clustering task oriented representation. However, the algorithm mainly focuses on the topology structure, such as the similarity between nodes, etc., which ignores the mesoscopic community structure information of the network.

Motivated by the above observations, we present an end-to-end network representation learning method for the task of community detection in this paper. To fully utilize the topological structure and node attributes, we develop the graph attention auto-encoder to get the initial representation. To fully capture higher-order community structure of the network, modularity maximization is presented. We further propose the self-training clustering module, which takes clustering assignments as soft labels to guide the representation learning.

Our contributions to this work are summarized as follows.

- We propose to jointly model topology structure and attribute information with the graph attention auto-encoder, and fuse both of them into the node representation with an encoder–decoder architecture.
- We present a framework of the unsupervised network representation learning for community detection. The framework is composed of three modules: graph attention auto-encoder, modularity maximization module and self-training clustering module. We formalize a unified objective function to optimize them simultaneously, which achieves the mutual promotion of the three parts.
- We implement experiments on three citation datasets and compare the performance of CDBNE with ten competitive baselines. The experiment results demonstrate the effectiveness of the proposed community detection model.

The following shows the organization of the remainder. Section 2 briefly concludes the related work. Section 3 defines the problems to be solved in this paper. We also illustrate the CDBNE model in detail. In Section 4, we evaluate CDBNE in community detection tasks and evaluate the network visualization aspect of the proposed method. Finally, we discuss this work and explore the future endeavors in Section 5.

2. Related work

Network representation Learning has recently gained tremendous attention due to its significant representational capabilities. Deepwalk (Perozzi et al., 2014) adopts random walk to sample the nodes and further learns the representations of the nodes by skip-gram. LINE (Tang et al., 2015) utilizes breadth-first search to expand the neighbors of nodes. It focuses on embedding large networks into the low-dimensional space while preserving both the first-order and second-order similarities (Tu et al., 2016). GraRep (Cao et al., 2015) obtains the efficient representation by matrix decomposition. Specially, it defines k step loss functions to integrate rich local information, which captures the structural properties of the network.

Network representation learning methods which consider both attribute information and topology structure have been popular over the last few years (Wang et al., 2021). TADW (Yang et al., 2015) incorporates auxiliary information into the network representation learning in the framework of matrix decomposition, resulting in a high-quality representation. MMDW (Tu et al., 2016) combines matrix decomposition with node labeling information to search for classification boundaries, resulting in the latent representation which is more favorable to classification tasks. Single-chromosome evolutionary (Pourabbasi et al., 2021) combines structural and content information to select the optimal content neighbor nodes for community detection in order to improve the value of the objective function D and the modularity of the network. HM-Modularity (Huang, Wang, & Chao, 2021) is proposed as a higher-order structural approach for multi-layer network community detection. A harmonic motif is designed to integrate slightly different higher-order structural information from all layers. LANE (Huang et al., 2017) first calculates the node label similarity matrix, structural similarity matrix and attribute similarity matrix of the network, then maps them separately, and finally transforms them into a new common low-dimensional space. However, the above approaches still have the following two limitations: (1) The cost of manually labeling data is very high; (2) They cannot effectively exploit the potential information between network topology structure and attribute in sparse networks.

The auto-encoder is an unsupervised neural network model which can effectively extract features from the input data. As a result, it plays a critical role in attribute network learning. For example, SDNE (Wang et al., 2016) and ANRL (Zhang et al., 2018) use a deep auto-encoder model to extract nonlinear characteristics of the high-dimensional matrix. DEC (Xie et al., 2016) is proposed to simultaneously learn node representation and cluster assignment based on the graph auto-encoder framework. GAE and VGAE (Kipf & Welling, 2016) utilize the graph convolutional network as the encoder for learning the node representation, and then optimize the representation by reconstructing the topological structure data with the help of an inner product decoder. The difference between GAE and VGAE is that the first constructs an auto-encoder while the last builds a variational auto-encoder. Unlike the two methods above, GATE rebuilds the topology structure and attribute values, and learns the node representation based on the graph attention auto-encoder. ANEM (Li et al., 2021) retains the microscopic proximity structure, the mesoscopic community structure and the node attributes information. Specifically, both the microscopic proximity structure and node attributes are decomposed by Nonnegative Matrix Factorization method. HCEMM (Huang, Wang, & Philip, 2021) utilizes a higher-order connectivity structure to enhance the intracommunity connection of each view. It can detect the underlying community structure in a multiview network while recovering missing edges. CDANE (Xu et al., 2022) introduces a framework of the attributed network that utilizes the matrix decomposition to jointly capture the topology and node attributes information. DAEGC (Wang et al., 2019) proposes a new goal-oriented framework that co-optimizes representation learning and network clustering, enabling the two components to benefit from each other. However, the algorithm ignores the microscopic structure of the community, which is not beneficial for fully exploring potential information.

3. Proposed method

3.1. Problem definition

Definition 1 (*Attributed Network*).(Falih et al., 2018) Given an attributed network $G = (V, E, X)$, where $V = \{v_i\}_{i=1,\dots,N}$ is a set of N nodes. $E = \{(v_i, v_j)|1 \le i, j \le N, i \ne j\}$ is a set of edges. $X = [x_1; \dots; x_N]$ is the attribute matrix of G, where x_i is the attribute vector of V_i. Let $A \in R^{N \times N}$ represents the adjacency matrix and is defined as:

$$A_{i,j} = \begin{cases} 1, & (v_i, v_j) \in E \\ 0, & (v_i, v_j) \notin E \end{cases} . \tag{1}$$

Problem 1 (*Attributed Network Representation Learning*). Given an attributed network $G = (V, E, X)$, attributed network representation learning aims to learn a mapping function Φ: $V \to R^d$, which converts each node v_i in G into a d-dimensional vector R^d, where $d \ll N$. N is the number of nodes in the network G. The function Φ should maintain the structural characteristics and attribute information of the network.

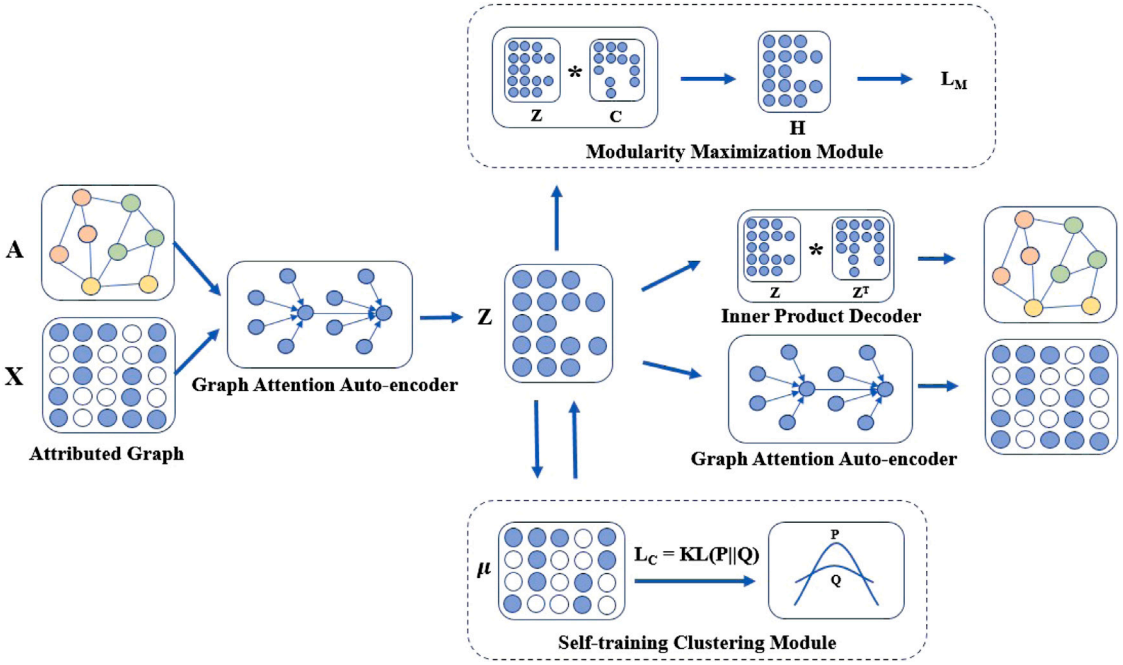

Fig. 1. The framework of the CDBNE model.

Problem 2 (*Community Detection*)**.** Given an attributed network G, community detection aims to identify K disjoint communities. The nodes in the same community are densely connected to each other and share similar features.

3.2. Overall framework

In this section, we propose CDBNE model for community detection on attributed networks. The overall framework is depicted in Fig. 1.

As can be shown, the framework consists of three parts, i.e., graph attention auto-encoder, modularity maximization module and self-training clustering module. **(1) Graph attention auto-encoder**: It obtains the representation by minimizing the loss of reconstructed topology and node attribute information. **(2) Modularity maximization module**: It exploits the similarity between higher-order nodes to optimize the node representation, which further captures the community structure of the original network. **(3) Self-training clustering module**: It combines network representation learning with community detection, enabling both components to benefit from each other, and generates the final node representation.

The main notations and relevant explanations are shown in Table 1.

3.3. Graph attention auto-encoder

Graph attention auto-encoder aims to embed attribute network into low-dimensional spaces while preserving the topology structure and attribute information as much as possible. This module contains two components: encoder and decoder. The encoder obtains the initial node representation by incorporating topology structure and attribute information of the network (Li et al., 2018). In addition, our decoder optimizes the node representation by reconstructing the original attribute network.

3.3.1. Encoder

To capture the network topology structure and the node attribute, the encoder updates new representation based on the correlations of the node with its neighbors. The encoder uses an attention mechanism to determine the correlation between v_i and its neighbor v_j, and their correlation coefficients at the lth encoder layer are set as Eq. (2).

$$r_{ij}^{(l)} = \sigma\left(v_s^{(l)^T}\phi\left(W^{(l)}z_i^{(l-1)}\right) + v_t^{(l)^T}\phi\left(W^{(l)}z_j^{(l-1)}\right)\right), \tag{2}$$

where, $W^{(l)} \in R^{d^{(l)}\times d^{(l-1)}}$, $v_s^{(l)} \in R^{d^{(l)}}$ and $v_t^{(l)} \in R^{d^{(l)}}$ are the weight parameters of the lth encoder layer, $z_i^{(l-1)} \in R^{d^{(l)}}$ denotes the representation of v_i on the lth encoder layer, σ and ϕ denote the activation function. We use the softmax function to normalize the obtained correlation coefficients as shown in Eq. (3).

$$\alpha_{ij}^{(l)} = \frac{\exp\left(r_{ij}^{(l)}\right)}{\sum_{t\in N_i}\exp\left(r_{it}^{(l)}\right)}, \tag{3}$$

where N_i denotes the neighborhoods of v_i. The calculation procedure for the representation of v_i on the lth encoder layer is shown in Eq. (4).

$$z_i^{(l)} = \sum_{j\in N_i}\alpha_{ij}^{(l)}\sigma\left(W^{(l)}z_j^{(l-1)}\right), \tag{4}$$

where $W^{(l)}$ is the weight matrix, σ represents the activation function. The initial node representation is learned by encoding the topology structure and attribute values through a two-layer stacked neural network.

3.3.2. Decoder

To invert the encoding process, we employ a decoder to reconstruct the resulting node representation. Similarly, the decoder updates the node representation based on the correlations among the nodes. To quantify the correlation between nodes and neighbors, we define the correlation coefficient between v_i and its neighbor v_j on the lth decoder layer, as given in Eq. (5).

$$\hat{e}_{ij}^{(l)} = \sigma\left(\hat{v}_s^{(l)^T}\phi\left(\hat{W}^{(l)}\hat{z}_i^{(l)}\right) + \hat{v}_t^{(l)^T}\phi\left(\hat{W}^{(l)}\hat{z}_j^{(l)}\right)\right), \tag{5}$$

where $\hat{W}^{(l)} \in R^{d^{(l)}\times d^{(l-1)}}$, $\hat{v}_s^{(l)} \in R^{d^{(l-1)}}$ and $\hat{v}_t^{(l)} \in R^{d^{(l-1)}}$ are the weight parameters of the lth decoder layer, $\hat{z}_i^{(l)} \in R^{d^{(l)}}$ denotes the representation of v_i in the lth decoder layer, and $d^{(l)}$ is the dimension of the representation, σ and ϕ denote the activation functions.

Table 1
Notations and explanations.

Symbols	Descriptions
G	The original attributed network.
V	The set of nodes in attributed network G.
A	The adjacency matrix of attribute network G.
X	The attribute matrix of network G.
m	The number of edges in network G.
$r_{ij}^{(l)}$	The correlation coefficient between v_i and v_j of the layer l encoder.
$\hat{r}_{ij}^{(l)}$	The correlation coefficient between v_i and v_j of the layer l decoder.
$W^{(l)}$, $v_s^{(l)}$, $v_t^{(l)}$	The weight parameters of layer l encoder.
$\hat{W}^{(l)}$, $\hat{v}_s^{(l)}$, $\hat{v}_t^{(l)}$	The weight parameters of layer l decoder.
$\alpha_{ij}^{(l)}$	The normalized correlation coefficient between v_i and v_j of layer l encoder.
$\hat{\alpha}_{ij}^{(l)}$	The normalized correlation coefficient between v_i and v_j of layer l decoder.
Z_i	The representation of v_i.
B	The modularity matrix.
d_i	The degree of v_i.
H	The temporary community assignment matrix.
C	The learnable fully connected layer matrix.
Q	The assignment distribution matrix.
P	The auxiliary target distribution matrix.
β	The hyperparameters for modularity maximization modules.
γ	The hyperparameters for self-training clustering modules.

The softmax function is adopted to normalize the resulting correlation coefficients as illustrated in Eq. (6).

$$\hat{\alpha}_{ij}^{(l)} = \frac{\exp\left(\hat{e}_{ij}^{(l)}\right)}{\sum_{t\in N_i}\exp\left(\hat{e}_{it}^{(l)}\right)}. \tag{6}$$

where N_i is the neighborhood of v_i. The final layer of the encoder provides results to the decoder. The lth decoder layer gets the representation of v_i in the previous layer according to Eq. (7).

$$\hat{z}_i^{(l-1)} = \sum_{j\in N_i}\hat{\alpha}_{ij}^{(l)}\sigma\left(\hat{W}^{(l)}\hat{z}_i^{(l)}\right), \tag{7}$$

where σ denotes the activation function, and $\hat{W}^{(l)}$ is the weight matrix.

The final result of the decoder is taken as the reconstructed node attribute values, i.e., $\hat{z}_i^{(0)}$ is the reconstructed attribute matrix $\hat{X}_i$ of v_i. The loss function of the reconstructed node attribute values is defined as in Eq. (8).

$$L_F = \sum_{i=1}^{N}\left\|X_i - \hat{X}_i\right\|_2^2, \tag{8}$$

where, X_i and $\hat{X}_i$ denote the original and reconstructed attribute matrix of v_i, respectively.

The inner product is used to calculate the cosine similarity of two vectors, which is useful when the size of the distance metric of the vectors is constant. Since potential representation already contains topology structure and node attribute, it would be efficient and flexible to use the simple inner product decoder. The loss function of predicting adjacency matrix is indicated in Eq. (9).

$$L_G = \sum_{j\in N_i}\phi\left(z_i^T z_j\right), \tag{9}$$

where ϕ is the activation function. The final reconstruction loss function is formed by combining the reconstruction loss of topology structure and node attribute, as illustrated in Eq. (10).

$$L_R = \frac{1}{N}\sum_{i=1}^{N}\left[\left\|X_i - \hat{X}_i\right\|_2^2 + \gamma\sum_{j\in N_i}\phi\left(z_i^T z_i\right)\right], \tag{10}$$

where γ is in charge of regulating the topological loss contribution.

3.4. Modularity maximization

In this section, the modularity is employed to optimize the network representation learning to ensure that the learned embedding retains the network's community structure. Modularity is a metric that indicates the strength of the network community structure (Agarwal & Kempe, 2008). The definition is shown in Eq. (11).

$$Q = \frac{1}{4m}Tr(H^T BH), \tag{11}$$

where $B \in R^{N\times N}$ denotes the modularity matrix, and $B_{ij} = A_{ij} - \frac{d_i d_j}{2m}$. $Tr(\cdot)$ denotes the trace of the matrix. A, d_i and m indicate the adjacency matrix, degree of v_i and network's edges count, respectively (Chen et al., 2014). $H \in R^{N\times K}$ denotes the community assignment matrix and is defined as:

$$H_{i,k} = \begin{cases} 1, & v_i \in C_k \\ 0, & v_i \notin C_k \end{cases}, \tag{12}$$

where C_k represents the jth community. Since maximizing modularity is an NP-hard problem, we employ a relaxed concept of the modularity, i.e., $Tr(H^T H) = N$. The community assignment matrix H is normalized as shown in Eq. (13).

$$\hat{H}_{ik} = \frac{\sqrt{N}\cdot\sqrt{H_{ik}}}{\sum_i\sum_k\sqrt{H_{ik}}}, \tag{13}$$

Each row of the matrix $\hat{H}$ is considered as a low-dimensional representation of the corresponding node, and then the modularity is used as an indicator to optimize the network representation learning. The node representation obtained based on this strategy captures low-level information of the network while maintaining the community structure of the original network, which illustrates the feasibility of optimizing the embedding by maximizing modularity (Yang et al., 2016).

Since the number of communities in a network is much smaller than the nodes, the representation learned is generally not enough to express the rich semantic information if the number of communities is used as the embedding dimension of nodes. A learnable fully connected layer $C \in R^{h\times k}$ is introduced to solve the problem. The final modularity maximization loss function is given as Eq. (14).

$$L_M = \frac{1}{4m}Tr\left(\hat{H}^T B\hat{H}\right) with\ \hat{H} = softmax(ZC). \tag{14}$$

3.5. Self-training clustering

We obtain the node representation by incorporating topological and graph attribute information, through a self-supervised mechanism and optimize them using the KL scattering objective function (Jin et al., 2019). The clustering loss function is shown in Eq. (15).

$$L_C = KL(P \parallel Q) = \sum_i \sum_u p_{iu} \log \frac{p_{iu}}{q_{iu}}, \tag{15}$$

where q_{iu} represents the probability that node v_i is assigned to community u. Given the initial node representation z_i, K-means clustering is applied to obtain the initial community center vector μ_u. Finally, the student's t-distribution is utilized to measure the similarity between z_i and μ_u. The formula is shown in Eq. (16).

$$q_{iu} = \frac{\left(1 + \|z_i - \mu_u\|^2/\alpha\right)^{-\frac{\alpha+1}{2}}}{\sum_k \left(1 + \|z_i - \mu_k\|^2/\alpha\right)^{-\frac{\alpha+1}{2}}}, \tag{16}$$

where $\alpha \geq 1$ represents the degree of freedom of the student's t-distribution, and $\alpha = 1$ was set in all experiments. The higher degree of freedom makes the target distribution P more centralized than Q. P assigns greater probabilities to similar nodes and lower probabilities to dissimilar nodes in the latent space. The target distribution of p_{iu} is defined as shown in Eq. (17).

$$p_{iu} = \frac{q_{iu}^2/(\sum_i q_{iu})^{\frac{1}{2}}}{\sum_k \left(q_{ik}^2/(\sum_i q_{ik})^{\frac{1}{2}}\right)}. \tag{17}$$

The target distribution P is identified by Q to emphasize the role of "confident assignment". The clustering loss forces Q to approach P, so both can supervise and optimize each other through continuous iterative updating. CDBNE obtains the low-dimensional embedding of nodes during the pre-training process. The embedding retains the low-order topology structure and the mesoscopic community structure of the original network through the reconstruction loss and modularity loss. The loss function is illustrated in Eq. (18).

$$L = L_R - \beta L_M, \tag{18}$$

The K-means method is applied to obtain the initial community centers needed for the self-training cluster optimization module during the pre-training. Then, the module optimizes the embedding and gets the final results for community detection. The final objective function of CDBNE is defined in Eq. (19).

$$L = L_R - \beta L_M + \gamma L_C, \tag{19}$$

where β and γ are hyperparameters that represent different loss weights. L_R, L_M and L_C denote the loss of the graph attention auto-encoder, modularity maximization and self-training clustering module, respectively. We utilize the final obtained optimized Q to obtain the node labels shown by Eq. (20).

$$S_i = \arg\max_u q_{iu}, \tag{20}$$

The main steps of CDBNE are given in Algorithm 1. Given N representing the number of nodes, E denoting the edges in graph, F being the number of node attributes, d_i being the dimension of the ith layer, D and k representing the maximum d_i in all layers and the number of clusters, respectively. The pre-training requires $O(k \log E + NFD + ED)$, the clustering optimization requires $O(Nd_ik)$. For Eq. (15), the time complexity is $O(Nk + N \log N)$. Thus, the total time complexity of Algorithm 1 is $O(T_1(k \log E + NFD + ED) + Nd_ik + T_2(Nk + N \log N))$.

Algorithm 1 Training process of CDBNE

Input: Graph $G = (A, X)$, number of communities K, hyperparameters β and γ, max number of pre-training iterations T_1, max number of training iterations T_2, target distribution update interval T
Output: Node representation Z, community center μ, node label S

```
for epoch = 1 : T_1 do
    Perform pre-training according to Eq. (18).
end for
Clustering on the node representation Z to get the initial community center μ.
for epoch = 1 : T_2 do
    if epoch%T == 0 then
        Calculate the target distribution P according to Eq. (17).
        Calculate node label S according to Eq. (20).
    end if
    for A and X do
        Calculate clustering loss L_C according to Eq. (15).
        Update the whole framework by minimizing Eq. (19).
    end for
end for
```

Table 2
Statistics of datasets.

Dataset	Nodes	Edges	Features	Communities
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3

4. Experiments

4.1. Datasets

We perform experiments on three real-world datasets, including Cora, Citeseer and Pubmed. The experimental datasets are briefly summarized in Table 2.

Cora[1]: The Cora dataset contains a total of 2,708 papers and 5,429 citation links between papers. All papers belong to 7 different machine learning fields, and each paper is represented by a keyword denoted by a 1,433-dimensional word vector.

Citeseer[1]: It is a subset of the Citeseer dataset. It contains 3,312 papers and 4,732 citation links between papers. All papers belong to 6 different academic research fields, and each paper is represented by a keyword denoted by a 3,703-dimensional word vector.

Pubmed[1]: It contains 19,717 papers and 44,338 citation links between papers. All papers belong to three different research fields, and each paper is identified by a keyword identified by a 500-dimensional word vector.

4.2. Evaluation metrics

Three commonly used metrics, namely accuracy (ACC), normalized mutual information (NMI) and adjusted rand index (ARI), are adapted to evaluate the performance of CDBNE.

ACC is the ratio of the number of correctly predicted samples to whole samples, which is defined as given in Eq. (21).

$$ACC(r, p) = \frac{\sum_{i=1}^{N} \delta\left(r_i, \text{map}\left(p_i\right)\right)}{N}, \tag{21}$$

where r_i is the actual category label of the ith sample, p_i is the predicted category label of the model on the ith sample. The $map()$ function facilitates each predicted category label to reach its optimal accurate

[1] https://linqs.soe.ucsc.edu/data

$$ARI(A,B)=\frac{\sum_{i=1}^{C_A}\sum_{j=1}^{C_B}\binom{N_{ij}}{2}-\left(2\cdot\sum_{i=1}^{C_A}\binom{N_i}{2}\sum_{i=1}^{C_B}\binom{N_j}{2}\right)/(N(N-1))}{\frac{1}{2}\left(\sum_{i=1}^{C_A}\binom{N_i}{2}+\sum_{i=1}^{C_B}\binom{N_j}{2}\right)-\left(2\cdot\sum_{i=1}^{C_A}\binom{N_i}{2}\sum_{i=1}^{C_B}\binom{N_j}{2}\right)/(N(N-1))}, \tag{25}$$

Box I.

category label. $\delta(x,y)$ denotes an indicator function defined as shown in Eq. (22):

$$\delta(x,y)=\begin{cases}1, x=y\\ 0, x\neq y\end{cases}. \tag{22}$$

NMI is the ratio between the classification result obtained by algorithm and the actual classification result. NMI has a range of 0 to 1 as its value. The better the algorithm performs, the higher the NMI value. Given the true and predicted category label r and p, the definition of NMI is shown in Eq. (23), where $H(r)$ and $H(p)$ denote the entropy of r and p, respectively. The definition of MI is shown in Eq. (24).

$$NMI(r,p)=\frac{2\cdot MI(r,p)}{H(r)+H(p)}, \tag{23}$$

$$MI(r,p)=\sum_{i=1}^{|r|}\sum_{j=1}^{|p|}p(r_i,p_j)\log_2\left(\frac{p(r_i,p_j)}{p(r_i)p(p_j)}\right), \tag{24}$$

where $p(r_i,p_j)$ is the probability that the selected sample belongs to both r_i and p_j, $p(r_i)$ is the probability that the randomly selected sample belongs to r_i.

ARI is used to evaluate the performance of method by calculating the similarity of the predicted and actual label distribution. Given the above label distributions of the sample, ARI is defined as shown in Eq. (25) (see Box I). Where $\binom{\cdot}{\cdot}$ is a binomial coefficient, A and B are two distributions of dataset C, and N_{ij} represents the number of objects located in both category i of partition A and category j of partition B. $N_{(i\cdot)}$ denotes the number of objects in category i of partition A, $N_{(j\cdot)}$ is the number of objects in category j of partition B. The value of ARI ranges from −1 to 1. The better the algorithm performs, the higher the ARI value.

4.3. Baselines

We compare CDBNE with ten competitive methods. The details are shown below.

K-means (Krishna & Murty, 1999): It initializes K distinct clusters, calculates the center of each cluster using the mean calculation method, and then iteratively updates the cluster center until the criterion function converges.

GraphEncoder (Salehi & Davulcu, 2020): It learns the nonlinear representation of the original network through a stacked auto-encoder, and achieves the clustering results by K-means method.

DeepWalk (Perozzi et al., 2014): It utilizes the random wandering strategy to collect the local information, and employs a neurolinguistic model to obtain the final embedding of the network structure.

TADW (Yang et al., 2015): It integrates node textual information into network representation learning via matrix decomposition, thus incorporating both rich topological structure and semantic information.

GAE&VGAE (Kipf & Welling, 2016): They integrate topology and attribute information values into the learned representation using a graph auto-encoder constructed by the graph convolutional network.

MGAE (Wang et al., 2017): It extends the graph auto-encoder to attribute graphs, and obtains the final node representation by minimizing the network adjacency matrix reconstruction loss.

ARGA&ARVGA (Pan et al., 2019): They employ a graph auto-encoder to extract the topology structure and node attribute values. They obtain more robust node representations by adversarial training mechanism.

DAEGC (Wang et al., 2019): It performs network clustering and the vector representation learning jointly in a unified framework. And it optimizes jointly auto-encoder reconstruction loss and clustering loss to simultaneously obtain both representation and clustering results.

GATE (Salehi & Davulcu, 2020): It uses stacked encoder/decoder layers with self-attention mechanisms to reconstruct inputs, including node attribute values and the graph topology.

SENet (Zhang et al., 2021): It enhances network structure by using shared neighbor information, and learns node embedding by using the spectral clustering loss. The model integrates topology structure and node attribute values into the kernel matrix by higher-order graph convolution network.

4.4. Experimental results on the task of community detection

In this section, we conduct community detection experiments to evaluate the performance of CDBNE on Cora, Citeseer and Pubmed. For the models without clustering methods such as DeepWalk, TADW and GATE, the K-means is used for detecting the community of the learned representation. We run CDBNE 10 times on each dataset to obtain the average value of ACC, NMI and ARI.

The details of experimental results are shown in Table 3, where the best values are in bold. ***A***, ***X*** and ***A&X*** indicate if the method utilizes only the network topology, attribute information, or both network topology and attribute information, respectively. CDBNE outperforms baselines methods on the three datasets. Specifically, it improves the NMI and ARI evaluation metrics on the Pubmed by 2.0% and 3.7% respectively, which verifies the effectiveness of CDBNE algorithm. From the results shown in the Table 3, we can derive following conclusions.

(1) TADW and GAE outperform K-means, Graph Encoder and DeepWalk methods. It illustrates that both topology structure and attribute information contains rich information.
(2) CDBNE outperforms SENet, which indicates the effectiveness of reconstructing network attribute information. Our method obtains higher quality representation by fully exploiting the topology structure and attribute information compared to SENet.
(3) CDBNE outperforms DAEGC, indicating the necessity of maintaining the mesoscopic community structure of the network and reconstructing the attribute information. DAEGC is superior to the CDBNE_mod (shown in the ablation study) which does not maintain the mesoscopic community structure. This indicates that our method optimizes the obtained node representation by using the modularity metric, which facilitates obtaining topologically tight community results.

4.5. Ablation study

We further validate the effectiveness of each module in CDBNE by ablation experiments. We define the following three variants:

- CDBNE_mod: CDBNE without the modularity maximization module. It indicates that the node representation is only optimized with the graph attention auto-encoder and self-training clustering module.

Table 3
The performance of CDBNE and competitors on community detection task.

Methods	Info	Cora			Citeseer			Pubmed		
		ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
K-means	**X**	0.500	0.547	0.501	0.544	0.312	0.285	0.580	0.278	0.246
Graph Encoder	**A**	0.301	0.059	0.046	0.293	0.057	0.043	0.531	0.210	0.184
DeepWalk	**A**	0.529	0.384	0.291	0.390	0.131	0.137	0.647	0.238	0.255
TADW	**A&X**	0.536	0.366	0.240	0.529	0.320	0.286	0.565	0.224	0.177
GAE	**A&X**	0.530	0.397	0.293	0.380	0.174	0.141	0.632	0.249	0.246
VGAE	**A&X**	0.592	0.408	0.347	0.392	0.163	0.101	0.619	0.216	0.201
MGAE	**A&X**	0.684	0.511	0.448	0.661	0.412	0.414	0.593	0.282	0.248
ARGA	**A&X**	0.640	0.449	0.352	0.573	0.350	0.341	0.681	0.276	0.291
ARVGA	**A&X**	0.638	0.450	0.374	0.544	0.261	0.245	0.513	0.117	0.078
DAEGC	**A&X**	0.704	0.528	0.496	0.672	0.397	0.410	0.671	0.266	0.278
GATE	**A&X**	0.658	0.527	0.451	0.616	0.401	0.381	0.673	0.322	0.299
ARVGA-AX	**A&X**	0.711	0.526	0.495	0.581	0.338	0.301	0.640	0.239	0.226
SENet	**A&X**	0.719	**0.550**	0.490	0.675	0.417	0.424	0.676	0.306	0.297
CDBNE	**A&X**	**0.728**	0.537	**0.513**	**0.697**	**0.438**	**0.455**	**0.707**	**0.336**	**0.337**

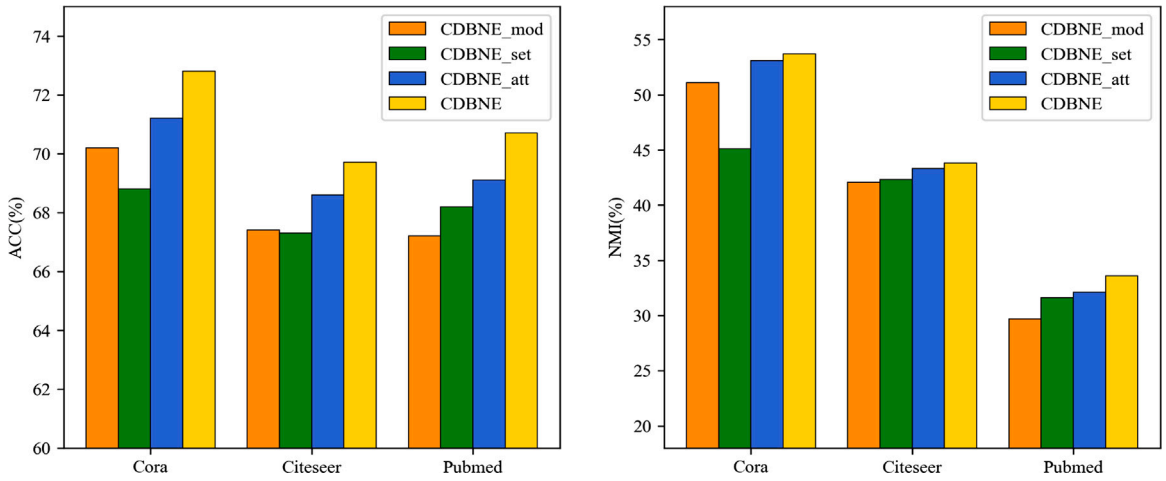

Fig. 2. The results of ablation study.

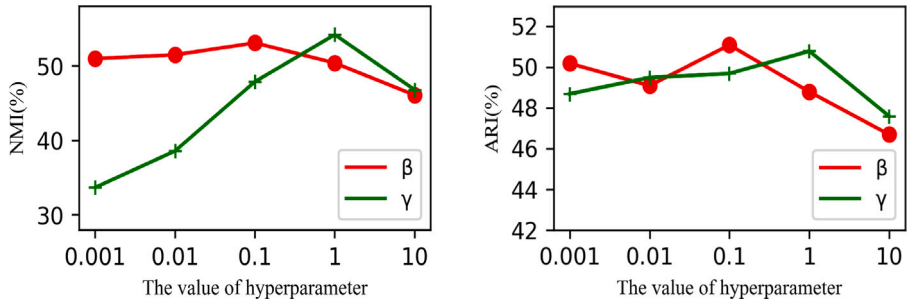

Fig. 3. Comparison of performance of CDBNE with different β and γ.

- CDBNE_set: CDBNE without the self-training clustering module. It specifies that the node representation is only optimized with the graph attention auto-encoder and modularity maximization module.
- CDBNE_att: CDBNE without the attribute decoder module. It indicates that the decoder only reconstructs the topology structure during the decoding.

The ablation experimental results are shown in Fig. 2, which clearly shows that each module contributes to the final performance. For example, CDBNE improves the ACC by 4.0% compared to CDBNE_set on Cora. It indicates that high quality representation can be obtained using the strategy of joint network representation learning and community detection. CDBNE improves the ACC by 1.6% compared to CDBNE_att on Cora, which proves the necessity of reconstructing node attribute values. CDBNE improves the ACC by 2.6% compared to CDBNE_mod on Cora, which validates the effectiveness of the modularity maximization module.

4.6. Parameters study

To study the sensitivity of β and γ, we select Cora as the experimental dataset. β and γ are used to control the weights of the modularity maximization and the self-training clustering module, respectively. In this section, the two parameters are tested in the $\{\mathbf{0.001}, \mathbf{0.01}, \mathbf{0.1}, \mathbf{1}, \mathbf{10}\}$ and evaluated by NMI and ARI.

As can be seen from Fig. 3, the scores of NMI and ARI increase and then decrease with the increase of β and γ values. On Cora, the best performance of the model is achieved when the values of β and γ are set to 0.1 and **1**, respectively. It illustrates the importance of balancing the

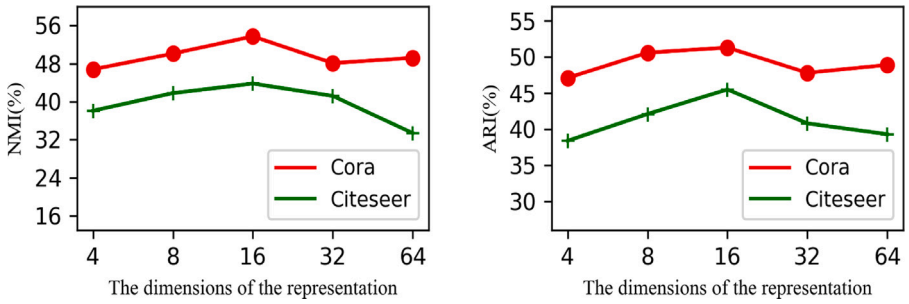

Fig. 4. Comparison of performance of CDBNE with different dimensions of the embedding.

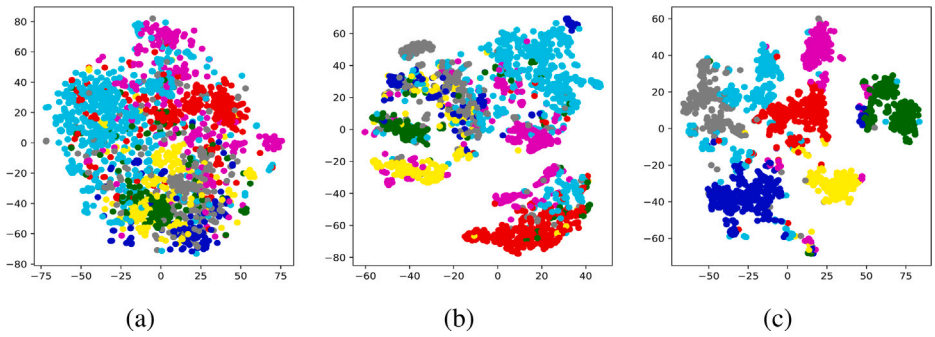

Fig. 5. Visualization of the CDBNE on Cora during training. (a) The initial visualization of the dataset. (b) The visualization on mid-training period. (c) The visualization of final results.

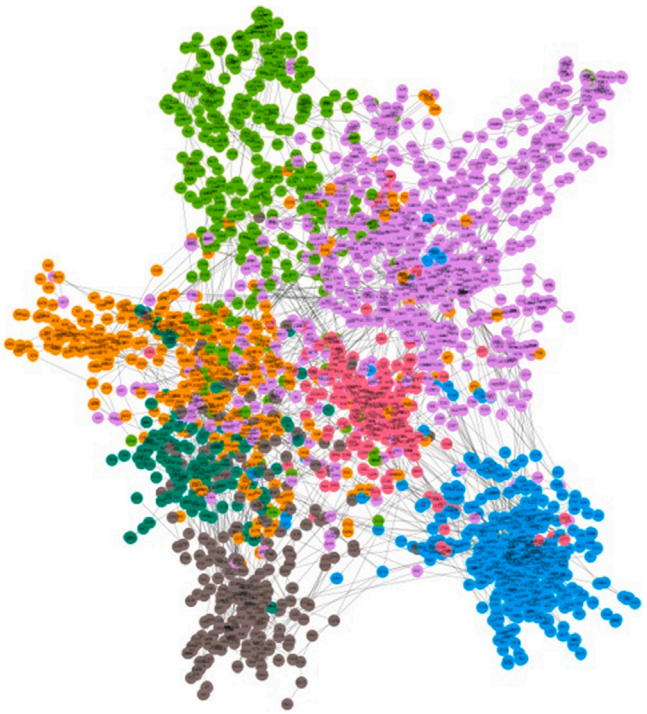

Fig. 6. Communities detected from the Cora dataset with the proposed CDBNE.

reconstruction network loss, the modularity maximization optimization loss and the self-training clustering loss. Similarly, the best result on Citeseer is obtained at "β=0.1, γ=1". In our community detection experiment, the most suitable values are chosen. To research the effect of the dimension of node representation on model performance, we set the values of β and γ to 0.1 and **1**. We set the range of the node representation dimension to $\{\mathbf{4, 8, 16, 32, 64}\}$. The experimental results are given in Fig. 4.

As demonstrated in Fig. 4, the performance of CDBNE shows a similar sensitivity to the dimensionality of the node representation on each dataset. The scores of NMI and ARI increase and then decrease as the dimensionality of the node representation increases. CDBNE achieves the best scores of NMI and ARI on Cora and Citeseer when the dimension of node representation increases to **16**. It indicates that selecting an appropriate node representation dimension is crucial. If the dimension of representation is too small, the implicit information of the network will be lost. Otherwise, the node representation learning will be interspersed with a lot of noise.

4.7. Visualization

In this section, we conduct two visualization experiments to show our results intuitively.

(1) We use the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm to map the learned vector representation to a two-dimensional space to demonstrate the effect of CDBNE more intuitively. Specifically, we visualize the representation obtained during the training on Cora, and the different colors of the nodes represent the community labels of the nodes.

The results are given in Fig. 5. In the initial stage, nodes of different types are intermingled, and the boundaries are not obvious, which improves as the training progresses. The node representation becomes evident until the final stage, with nodes in the same community tightly clustered and nodes in different communities with distinct boundaries. In addition, there are few isolated nodes or small groups of nodes.

(2) We visualize the detected communities on Cora, which is given in Fig. 6. The number on the vertice represents its own ID, and the color corresponds to different community labels. The NMI is 0.537. According to Fig. 6, we can see that the CDBNE clearly divides the network into seven communities. It proves that CDBNE provides us good clustering results and fully explores the community structure.

5. Conclusion

In this paper, we present an unsupervised community detection method based on attribute network representation learning (CDBNE). We propose a unified objective function to combine clustering loss and embedding loss to ensure that the embedding is suitable for the community detection task. Thus, it can efficiently map topology structure and attribute information into a low-dimensional space by CDBNE, while maximally preserving the clustering-oriented information. Experimental results show that CDBNE outperforms state-of-the-art methods on several datasets of the community detection task. As our proposed method is designed for the static attributed network, which does not apply in the dynamic attributed network. In addition, it only applies to non-overlapping community detection where one node belongs to only one community. Therefore, in the future we will research embedding approaches for the dynamic attributed network and consider overlapping community detection.

CRediT authorship contribution statement

Xinchuang Zhou: Investigation, Writing – original draft, Data curation. **Lingtao Su:** Methodology, Writing – review & editing. **Xiangju Li:** Conceptualization, Investigation. **Zhongying Zhao:** Methodology, Supervision, Writing – review & editing, Funding acquisition. **Chao Li:** Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The source codes have been shared with a link in abstract.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant No. 62072288, 61702306, 61433012), the Taishan Scholar Program of Shandong Province (Grant No. ts20190936), the Natural Science Foundation of Shandong Province, China (Grant No. ZR2018BF013, ZR2022MF268), the National Key R&D Plan, China (Grant No. 2018YFC0831002), the Key Project of Industrial Transformation and Upgrading in China (Grant No. TC170A5SW).

References

Agarwal, G., & Kempe, D. (2008). Modularity-maximizing graph communities via mathematical programming. *The European Physical Journal B*, *66*(3), 409–418.

Cao, S., Lu, W., & Xu, Q. (2015). Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management* (pp. 891–900).

Cao, S., Lu, W., & Xu, Q. (2016). Deep neural networks for learning graph representations. In *Proceedings of the 30th AAAI conference on artificial intelligence* (pp. 1145–1152).

Chen, M., Kuzmin, K., & Szymanski, B. K. (2014). Community detection via maximization of modularity and its variants. *IEEE Transactions on Computational Social Systems*, *1*(1), 46–65.

Doluca, O., & Oğuz, K. (2021). APAL: Adjacency propagation algorithm for overlapping community detection in biological networks. *Information Sciences*, *579*, 574–590.

Falih, I., Grozavu, N., Kanawati, R., & Bennani, Y. (2018). Community detection in attributed network. In *Companion proceedings of the the web conference 2018* (pp. 1299–1306).

Gao, Y., Yu, X., & Zhang, H. (2021). Overlapping community detection by constrained personalized PageRank. *Expert Systems with Applications*, *173*, Article 114682.

Huang, X., Li, J., & Hu, X. (2017). Label informed attributed network embedding. In *Proceedings of the 10th ACM international conference on web search and data mining* (pp. 731–739).

Huang, L., Wang, C.-D., & Chao, H.-Y. (2021). HM-Modularity: A harmonic motif modularity approach for multi-layer network community detection. *IEEE Transactions on Knowledge and Data Engineering*, *33*(6), 2520–2533. http://dx.doi.org/10.1109/TKDE.2019.2956532.

Huang, L., Wang, C.-D., & Philip, S. Y. (2021). Higher order connection enhanced community detection in adversarial multiview networks. *IEEE Transactions on Cybernetics*.

Jin, D., Li, B., Jiao, P., He, D., & Shan, H. (2019). Community detection via joint graph convolutional network embedding in attribute network. In *International conference on artificial neural networks* (pp. 594–606).

Jin, D., Yu, Z., Jiao, P., Pan, S., He, D., Wu, J., Yu, P., & Zhang, W. (2021). A survey of community detection approaches: From statistical modeling to deep learning. *IEEE Transactions on Knowledge and Data Engineering*.

Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. arXiv preprint arXiv:1611.07308.

Krishna, K., & Murty, M. N. (1999). Genetic K-means algorithm. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *29*(3), 433–439.

Li, J.-H., Huang, L., Wang, C.-D., Huang, D., Lai, J.-H., & Chen, P. (2021). Attributed network embedding with micro-meso structure. *ACM Transactions on Knowledge Discovery from Data*, *15*(4), 1–26.

Li, F., Qiao, H., & Zhang, B. (2018). Discriminatively boosted image clustering with fully convolutional auto-encoders. *Pattern Recognition*, *83*, 161–173.

Liu, F., Xue, S., Wu, J., Zhou, C., Hu, W., Paris, C., Nepal, S., Yang, J., & Yu, P. S. (2020). Deep learning for community detection: Progress, challenges and opportunities. arXiv preprint arXiv:2005.08225.

Mishra, S., Singh, S. S., Mishra, S., & Biswas, B. (2021). TCD2: Tree-based community detection in dynamic social networks. *Expert Systems with Applications*, *169*, Article 114493.

Nadimi-Shahraki, M. H., Moeini, E., Taghian, S., & Mirjalili, S. (2021). DMFO-CD: A discrete moth-flame optimization algorithm for community detection. *Algorithms*, *14*(11), 314.

Naik, D., Ramesh, D., Gandomi, A. H., & Gorojanam, N. B. (2022). Parallel and distributed paradigms for community detection in social networks: A methodological review. *Expert Systems with Applications*, *187*, Article 115956.

Pan, S., Hu, R., Fung, S.-f., Long, G., Jiang, J., & Zhang, C. (2019). Learning graph embedding with adversarial training methods. *IEEE Transactions on Cybernetics*, *50*(6), 2475–2487.

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* (pp. 701–710).

Pourabbasi, E., Majidnezhad, V., Afshord, S. T., & Jafari, Y. (2021). A new single-chromosome evolutionary algorithm for community detection in complex networks by combining content and structural information. *Expert Systems with Applications*, *186*, Article 115854.

Salehi, A., & Davulcu, H. (2020). Graph attention auto-encoders. In *Proceedings of the 32nd international conference on tools with artificial intelligence* (pp. 989–996).

Shao-hai, L., Jin-zhao, W., & Na, A. (2012). A algorithm based on the local module degree for community detection in complex networks. In *2012 IEEE international conference on computer science and automation engineering* (pp. 232–236).

Sun, G., & Zhang, X. (2019). A novel framework for node/edge attributed graph embedding. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 169–182).

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web* (pp. 1067–1077).

Tu, C., Zhang, W., Liu, Z., & Sun, M. (2016). Max-margin deepwalk: Discriminative learning of network representation. In *Proceedings of the 25th international joint conference on artificial intelligence* (pp. 3889–3895).

Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1225–1234).

Wang, X., Li, J., Yang, L., & Mi, H. (2021). Unsupervised learning for community detection in attributed networks based on graph convolutional network. *Neurocomputing*, *456*, 147–155.

Wang, C., Pan, S., Hu, R., Long, G., Jiang, J., & Zhang, C. (2019). Attributed graph clustering: A deep attentional embedding approach. In *Proceedings of the 28th international joint conference on artificial intelligence*.

Wang, C., Pan, S., Long, G., Zhu, X., & Jiang, J. (2017). Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 889–898).

Xie, J., Girshick, R., & Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *International conference on machine learning* (pp. 478–487).

Xu, X.-L., Xiao, Y.-Y., Yang, X.-H., Wang, L., & Zhou, Y.-B. (2022). Attributed network community detection based on network embedding and parameter-free clustering. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *52*(7), 8073–8086.

Yang, L., Cao, X., He, D., Wang, C., Wang, X., & Zhang, W. (2016). Modularity based community detection with deep learning. In *Proceedings of the 25th international joint conference on artificial intelligence, vol. 16* (pp. 2252–2258).

Yang, C., Liu, Z., Zhao, D., Sun, M., & Chang, E. (2015). Network representation learning with rich text information. In *Proceedings of the 24th international joint conference on artificial intelligence*.

Zhang, X., Liu, H., Wu, X.-M., Zhang, X., & Liu, X. (2021). Spectral embedding network for attributed graph clustering. *Neural Networks*, *142*, 388–396.

Zhang, Z., Yang, H., Bu, J., Zhou, S., Yu, P., Zhang, J., Ester, M., & Wang, C. (2018). ANRL: Attributed network representation learning via deep neural networks. In *Proceedings of the 27th international joint conference on artificial intelligence, vol. 18* (pp. 3155–3161).