Contents lists available at ScienceDirect

Expert Systems With Applications

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

A novel network core structure extraction algorithm utilized variational autoencoder for community detection

Rong Fei[a,b,*], Yuxin Wan[a], Bo Hu[c,**], Aimin Li[a,b], Qian Li[a,b]

[a] *School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, Shaanxi, PR China*
[b] *Shaanxi Key Laboratory for Network Computing and Security Technology, Xi'an, Shaanxi, PR China*
[c] *Hangzhou HollySys Automation Co., Ltd., Hangzhou, Zhejiang, PR China*

ARTICLE INFO

Dataset link: https://github.com/PeterWana/CSEA

Keywords:
Community detection
Complex networks
Core structure
Variational autoencoder
Clustering

ABSTRACT

Community detection technologies have the general research significance in complex networks, in which the topology information of network is worthy to be the focus for its widely application. It is the definition of community structure that the connection of nodes in the community is dense with the connection of nodes outside the community is sparse, which is corresponding to the core structure in the complex real networks is represented by a compact and dense set of connected nodes. While all the notes in the network are considered by the traditional topology, it is hard to extract the core structure with the continuous, exponential growth of community networks. In this paper, a novel network core structure extraction algorithm utilized variational autoencoder for community detection(CSEA) is proposed for finding the community structure more accurately. Firstly, the K-truss algorithm is used to find the core structure information in the network, and the similarity matrix is generated by similarity mapping combined with local information. Secondly, the variational autoencoder is used to extract and reduce the dimension of the similarity matrix containing the core structure of the network, and the low-dimensional feature matrix is obtained. Finally, the K-means clustering algorithm is utilized to obtain the community structure information. We compare CSEA algorithm with 18 different types of community detection algorithms using 4 evaluation metrics on 19 complex real networks. By extensively evaluating our algorithm on large real-world datasets, we show that CSEA algorithm has an excellent community division effect in dense complex real networks, especially in small and medium-sized networks, and it can accurately divide the complex real networks with unknown community structure. Simultaneously, CSEA algorithm also reveals some efficiency advantage in its on-line test.

1. Introduction

The definition of communities is a class of sets of nodes with the same characteristics, and different communities represent clusters of different groups in the network. There are currently many types of community structures that form corresponding complex network whose origins can be includes chemistry, biology and sociology (Girvan & Newman, 2002). With the wide application of complex network, there are often a variety of complex connections that form corresponding networks as protein information networks (Tian, Gao, Cui, Chen, & Liu, 2014; Xin, Wang, Ying, & Wang, 2017) and social networks (Freeman, 2004; Jia, Cai, Musolesi, Wang, Tennant, Weber, Heath, & He, 2012). Communities show certain operational advantages in exploring the structure and function of the whole network, which may assist us to analyze or predict the interactions between entities in the network (Newman & Peixoto, 2015). Therefore, community detection techniques that are used to disclosure community structures in real networks came into being. Community detection (Fortunato & Hric, 2016) reveal the topology and network characteristics of community structures that interact with each other and help us to better understand and analyze real networks. In expert systems, community detection techniques also become important. We take the group recommendation as the example. The corresponding network feature representations to obtain different types of user divisions can be learning in social networks, which is good for achieving accurate ad placement and partition recommendations according to different user needs (Gao, Wu, Qiao, Zhou, Yang, & Hu, 2016; Liu, Zhou, Wu, Hu, & Guo, 2018), while community detection for users in e-commerce can help them build more

* Corresponding author at: School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, Shaanxi, PR China.
** Corresponding author.
E-mail addresses: annyfei@xaut.edu.cn (R. Fei), 3170913002@stu.xaut.edu.cn (Y. Wan), jobobo@outlook.com (B. Hu), liaiminmail@gmail.com (A. Li), qianli_30@163.com (Q. Li).

https://doi.org/10.1016/j.eswa.2023.119775
Received 13 May 2022; Received in revised form 24 January 2023; Accepted 27 February 2023
Available online 8 March 2023

reliable recommendation systems (Reddy, Kitsuregawa, Sreekanth, & Rao, 2002; Wu, Wang, Shafiee, & Zhang, 2021).

Traditional community detection algorithms focus on the network topology and the connections between nodes (Javed, Younis, Latif, Qadir, & Baig, 2018; Kim & Lee, 2015), which is usually represented by the corresponding adjacency matrix. The interrelation and constraint between nodes provide clues for community detection. However, in view of the properties of the real complex networks, there often exists a kind of special core structure, which plays a key role in complex networks analysis. For instance, in social networks, such a structure generally has an influential power or is an extremely active group. The core structure manifests itself as a compact and dense set of nodes connected to each other, which is closer to the basic definition of community structure (Girvan & Newman, 2002). There are a plenty of algorithms which can find the core structure in the real complex networks, such as clique (Bron & Kerbosch, 1973), K-core (Kong, Shi, Wu, & Zhang, 2019; Panduranga, Gao, Yuan, Stanley, & Havlin, 2017), K-truss (Zheng, Ye, Li, Ling, & Jin, 2017), K-plex (Berlowitz, Cohen, & Kimelfeld, 2015), etc. Among them, clique and K-plex have strict requirements for subnetworks, which take a long time in large networks, and the range of applications may have certain boundedness. Relatively, K-core and K-truss have higher efficiency, but K-truss is more cohesive than K-core. Moreover, they share one disadvantage in common, it seems more difficult and less obvious to find the core structure in medium and large sparse networks cause that they only analyze the core structure on the basis of the connections between nodes.

Another approach to perform community detection is to learn a new network representation from the network topology (Cavallari, Zheng, Cai, Chang, & Cambria, 2017; Huang, Dou, & Ma, 2021). Mapping the relational data from the original space to a low-dimensional feature space to achieve an effect of reducing the noise while preserving the information of the original network structure. Recently, community detection algorithms have also in combination with deep learning models, such as convolutional neural network (CNN)-based algorithms (De Santo, Galli, Moscato, & Sperlì, 2021; Sperlí, 2019), generative adversarial network (GAN)-based algorithms (Wang, Cao et al., 2021; Yang, Wang, Gu, Wang, Cao, & Guo, 2020), graph convolutional network (GCN)-based algorithms (Cui, Zhou, Yang, & Liu, 2020; De Santo et al., 2021; Fettal, Labiod, & Nadif, 2022; He, Song, Jin, Feng, Zhang, Yu, & Zhang, 2021; Jin, Li et al., 2019; Jin, Liu et al., 2019; Wang, Li et al., 2021), graph attention network (GAT)-based algorithms (Luo, Fang, Cao, Zhang, & Zhang, 2021; Park, Kim, Han, & Yu, 2020; Wang, Liu et al., 2021; You, Cho, Shon, Seo, & Kim, 2022) and autoencoder (AE)-based algorithms (Choong, Liu, & Murata, 2020; Fei, Sha, Xu, Hu, Wang, & Li, 2020; He et al., 2021; Jin, Ge, Li, Lu, He, & Fogelman-Soulie, 2017; Xie, Wang, Jiang, & Xu, 2019; Xu, Che, Wang, Hu, & Xie, 2020; Yang, Cao, He, Wang, Wang, & Zhang, 2016), etc., where neural networks are able to solve some problems that are too complex or where there are no human experts and no rules. Due to neural networks can be easier modification and much faster efficiency, combining neural networks and expert systems are more powerful than single expert systems or neural networks. Subsequently, the community division results will be more accurate. Considering the high-dimensional features of complex networks, autoencoders, a kind of neural networks, usually used to reduce the dimension and extract the features to obtain better results. We try to use a special autoencoders, namely variational autoencoder (Kingma & Welling, 2013), which is built on the basis of neural networks and combines the KL divergence and cross-entropy loss function to train the neural network parameters. Compared with other autoencoders, the variational autoencoder is rarely applied in community detection, and it can be applied to large-scale network datasets and overcome the problems caused by "dimension disaster". At the same time, the trained variational autoencoder is efficient and easy to modify, which is difficult to achieve with community detection algorithms based on logistic analysis techniques.

Specifically, we propose a novel network **core structure extraction algorithm(CSEA)** utilized variational autoencoder for community detection. Firstly, we utilize the K-truss algorithm to find the core structure in the network, subsequently the similarity matrix is obtained by similarity mapping considering with local information and core structure information. Secondly, the similarity matrix with the core structure information is subjected to feature extraction and dimension reduction by the variational autoencoder model. Finally, the community division is gained by the K-means algorithm. Thus, the main contributions of our work can be summarized as follows:

1. A new variational autoencoder model for community detection is designed on the basis of unsupervised learning methods, which is used to perform feature extraction and feature dimension reduction on the similarity matrix to obtain a low-dimensional feature matrix, and from the experimental results, we can conclude that our model can effectively analyze the community structures in large-scale networks and accelerate the clustering process compared to traditional logical analysis techniques.
2. A novel structure for CSEA is proposed in this paper, which combines K-truss algorithm and variational autoencoder model to accelerate the search speed of the core structure in the network. The core structure of the network is found by using the K-truss Algorithm, the similarity matrix is generated by similarity mapping and then fed into the variational autoencoder, and finally the community division is gained by the K-means algorithm.
3. The CSEA is tested on 19 real network datasets, which is compared with 18 different types of community detection algorithms on 4 evaluation metrics. The experimental results demonstrate that the CSEA algorithm has a better effect in dense complex networks for community division, especially in small and medium-sized networks. Meanwhile, CSEA algorithm also shows some efficiency advantage in its on-line test.

The rest of this paper is organized as follows: the related work and the relevant community detection algorithms are described in Section 2. The specific procedures of the CSEA algorithm are illustrated in Section 3. Experiments and results analysis of the CSEA algorithm are introduced in Section 4. Conclusion and future work are provided in Section 5.

2. Related work

Community detection algorithms have attracted the attention of many scholars from different research fields. For different problems, many methods based on different theories and technologies have been proposed (Su, Xue, Liu, Wu, Yang, Zhou, Hu, Paris, Nepal, Jin, et al., 2022). For example, the method based on graph theory, such as the Kernighan Lin (KL) (Kernighan & Lin, 1970) algorithm, is an effective dichotomy through a local search strategy. The method based on hierarchical clustering, such as the Girvan–Newman algorithm (GN) (Girvan & Newman, 2002), is used to discover the community in the graph by iteratively removing the edge with the maximum edge betweenness relative to the source node in the graph. Newman proposed another algorithm called the FastNewman algorithm (FN) (Newman, 2004), which is based on the GN algorithm. This algorithm can get a structure similar to the GN algorithm, but its time complexity is lower. The Clauset–Newman–More algorithm (CNM) (Clauset, Newman, & Moore, 2004) is a new greedy algorithm based on the above two algorithms. Clause et al. used the data structure of heaps to calculate and update the modularity of the network. All the above three community detection algorithms use the concept of modularity (Clauset et al., 2004; Newman & Girvan, 2004), while a more widely used algorithm based on modularity optimization is the fast unfolding algorithm (FUA) (Blondel, Guillaume, Lambiotte, & Lefebvre, 2008). The advantage of the FUA algorithm is that it is fast and can be used to achieve community division of large-scale networks at different granularities in a relatively short time without specifying the number of communities. The clustering method based on skeleton graphs, such as the gCluSkeleton algorithm (Huang, Sun, Song, Deng, & Han, 2012), is used to generate skeleton graphs according to the network topology, aggregate the structure information contained in the network into the backbone network, and extract the effective community structure from the complex connection relationship. Based on the label propagation method, for instance, the label propagation algorithm (LPA) (Raghavan, Albert, & Kumara, 2007), a unique label is initialized for each node in the network. During iteration, the node makes corresponding changes according to the label of the node connected to it and the division of the community is completed by using label propagation. When using the speaker–listener label propagation algorithm (SLPA) (Xie, Szymanski, & Liu, 2011), which is an extension of the LPA algorithm, each node is considered to have more than one tag, each iteration adds a tag, Finally, filtering occurs. Recently, with the integration and development of research, plenty of more novel community detection algorithms have been proposed. Example include, the Infomap algorithm (Rosvall & Bergstrom, 2008) combined with the information coding and random walks method, the mathematical programming method (Srinivas & Rajendran, 2019), the Relevance-based Information Interaction Model (RIIM) (Ullah, Wang, Sheng, Long, Khan, & Ejaz, 2022) combined with the local and global information of nodes, the divide and agglomerate algorithm (DAA) (Liu & Ma, 2019) combined with the novel hierarchical clustering method, the algorithm combined with the modularity and K-plexes method (MOKP) (Zhu, Chen, & Zeng, 2020), the method combined with modularity and the Markov random field (ModMRF) (Jin, Zhang, Song, He, Feng, Chen, Li, & Musial, 2020). The parallel label diffusion and label selection-based algorithm (PLDLS) (Roghani, Bouyer, & Nourani, 2021), the node importance based label propagation algorithm (NI-LPA) (El Kouni, Karoui, & Romdhane, 2020) and the graph layout based label propagation algorithm (GLLPA) (Zhang, Liu, Jin, Tao, Chen, & Wu, 2020) integrated label propagation algorithm with other novel methods, the parallel and distributed methods (Khomami, Rezvanian, & Meybodi, 2016; Naik, Ramesh, Gandomi, & Gorojanam, 2022), the methods combined with objective optimization and evolutionary strategy (Bara'a, Rada, Abbas, & Özdemir, 2019), and methods used for dynamic network community detection (Li, Zhu, Li, Wang, Dai, Wang, & Jin, 2021; Mohammadmosaferi & Naderi, 2020).

In addition, the community detection method based on network embedding is a popular community detection method. The basic idea of the network embedding method is to convert the nodes in the network into low-dimensional vectors, and require the vectors to reflect some features in the network, to obtain the community structure. In typical network embedding methods such as the Leading eigenvector algorithm (LE) (Newman, 2006a), the core of the algorithm is to define a modularity matrix. The role of the modularity matrix in community detection is similar to the role played by the graph Laplacian operator in graph partitioning. In the Spectral algorithm (SC) (Amini, Chen, Bickel, & Levina, 2013), which represents the network by an adjacency matrix, the eigenvalues and eigenvectors of the matrix are studied to solve the community detection problem. In nonnegative matrix factorization methods, such as the original nonnegative matrix factorization algorithm (NMF) (Gillis & Glineur, 2012), the basic principle is to factorize the original matrix into a community indicator matrix and base matrix, randomly initialize the elements of these two matrices, and iteratively update these two matrices until the objective function converges. Similarly, there are the SNMF (Wang, Li, Wang, Zhu, & Ding, 2011), MNDP (Jin, Chen, He, & Zhang, 2015), IDPCNMF (Lu, Shen, Sang, Zhao, & Lu, 2020) and so on. A typical network embedding method, such as the DeepWalk algorithm (Perozzi, Al-Rfou, & Skiena, 2014), based on random walks, which is used to learn the social representation of the network by combining the truncated random walk method and hidden information of the network. Good scalability and parallelism, can be achieved using this method. In addition, this method can perform well given high-dimensional sparse vectors. The node2vec algorithm (Grover & Leskovec, 2016) is an enhanced version of the DeepWalk algorithm. By adjusting the random walks weight, the result of network embedding is balanced between network homophily and structural equivalence. Similar improved algorithms include LINE (Tang, Qu, Wang, Zhang, Yan, & Mei, 2015), struct2vec (Ribeiro, Saverese, & Figueiredo, 2017) and SNS (Lyu, Zhang, & Zhang, 2017). In addition, Jin, You et al. (2019) incorporating the network embedding into the Markov random field model to obtain better community division. Lu et al. (2020)combined the improved density peak clustering method and the nonnegative matrix factorization method to solve the problem that NMF-based community detection methods require the number of communities as a priori information. Huang et al. (2021) proposed a nonnegative matrix factorization algorithm for multilayer network reduction (NMF-MNR), which combines topology and network representation of each layer and provides a better method for the feature description of each layer of the network.

With the wider machine learning technology, the community detection method based on network embedding can also be well combined with machine learning technology. Using the dimension reduction method, which is used to extract the main feature components of the data, and combining it with the common clustering algorithm for community division is also an idea of the network embedding method. General dimension reduction method such as principal component analysis (PCA) (Shlens, 2014), clustering methods such as the K-means algorithm (Bahmani, Moseley, Vattani, Kumar, & Vassilvitskii, 2012) and the Markov clustering algorithm (MCL) (Ochieng, Kusuma, & Haryanto, 2017). Another idea is to use neural network to extract the feature of the network. For example, structural deep network embedding model (SDNE) (Wang, Cui, & Zhu, 2016) uses an autoencoder to optimize a network embedding algorithm with similarity, and the vector obtained by learning can retain global and local network structure information. Similar algorithms include the DNGR (Cao, Lu, & Xu, 2016) and ANRL (Zhang, Yang, Bu, Zhou, Yu, Zhang, Ester, & Wang, 2018). Next to that, deep learning technology was introduced into the community detection. The current mainstream deep learning-based community detection methods are the algorithms by utilizing autoencoder or convolutional network. The first are the autoencoder-based (AE) community detection methods, Yang et al. firstly proposed the deep nonlinear reconstruction model (DNR) (Yang et al., 2016), in which the modularity matrix and autoencoder are combined. The modularity matrix is fed into the autoencoder, euclidean distance and sigmoid cross entropy are taken as loss function to train the network. Finally, the K-means algorithm is utilized to gain the community division. Jin et al. (2017) improved the DNR algorithm by combining the Laplacian matrix and modularity matrix. Choong et al. (2020) proposed a community detection method for a variational graph autoencoder based on dual optimization, which introduced lower bound maximization in the variational autoencoder and followed the optimization process of neural expectation maximization to ensure optimal community division. Fei et al. (2020) proposed a community detection algorithm based on a deep sparse autoencoder (DSACD) by combining node similarity and a deep sparse autoencoder model. Similarly, inspired by the relationship between nodes, Xie et al. (2019) designed a new and effective network adjacency matrix transformation method to describe the similarity of nodes in the network topology, and then, they proposed another new community detection algorithm based on the deep transitive autoencoder (CDDTA). Based on this research, Xu et al. (2020) combined the similarity of 4 different complex networks with stacked autoencoders to compensate for the deficiency of a single similarity matrix in describing the similarity relationship between nodes. The next are the convolutional network-based community detection methods, which contain convolutional neural network

(CNN)-based and graph convolutional network (GCN)-based community detection methods. Sperlí (2019) proposed a novel convolution technique particularly useful for sparse matrices to perform community detection. Jin et al. (2017) proposed an unsupervised community detection model based on graph convolutional neural network embedding. De Santo et al. (2021) proposed an innovative semi-supervised community detection method using convolutional neural networks to divide communities by using different properties, such as topology and context information of the network. Wang, Li et al. (2021) integrated a label sampling model and a graph convolutional neural network into an unsupervised learning framework to discover underlying community structures by fusing topology and attribute information. Moreover, autoencoder and convolutional network are combined in some method. Typically, under the framework of the autoencoder, He et al. (2021) proposed a new unsupervised community detection method combining a Markov random field and a graph convolution neural network. In addition, there are some algorithms exploiting on the basis of other neural networks. For instance, Yang et al. (2020) proposed a new Joint Adversarial Network Embedding (JANE) framework based on generative adversarial network(GAN) to jointly distinguish between real and fake combinations of embeddings, topological information and node features. Park et al. (2020) proposed a method called unsupervised attribution multiplexing network embedding (DMGI) to model the global properties of a graph by using an attention mechanism to infer that the importance of each relationship type maximizes the mutual information between local patches of the graph and the global representation of the entire graph. All the above algorithms have their own advantages and disadvantages. For different types of network community structures, the corresponding types of algorithms can often obtain better community division. Based upon the above idea, we propose a novel network core structure extraction algorithm utilized variational autoencoder for community detection.

3. Proposed method

In this section, the proposed CSEA algorithm is completely described in detail. First, the basic concepts and definitions are described. Then, we introduce the framework of CSEA algorithm.

3.1. Preliminaries

In this paper, the definition of network structure is as follows:

Local information: In an undirected network $G = (V, E)$, V represents a set of nodes E represents a set of edges, $i, j \in V, e_{ij} \in E, |V| = n, |E| = m$ and G is represented by an adjacency matrix $A = [a_{ij}]_{n\times n} \in R^{n\times n}$, which is defined as local information.

For local information A, if node i and node j are connected, $a_{ij} = 1$, otherwise $a_{ij} = 0$. When $a_{ij} = 1$, e_{ij} is defined as the connected edge between node i and node j, and $deg(i)$ is defined as the degree of node i.

Community information: In an undirected network $G = (V, E)$, define matrix $C'_p = [c'_i]_{n\times 1} \in R^{n\times 1}$ as community information and p as the number of different kinds of community labels obtained by different clustering algorithms.

Real community information: In an undirected network $G = (V, E)$, define matrix $C_q = [c_i]_{n\times 1} \in R^{n\times 1}$ as real community information and q as the number of different kinds of real community labels.

3.1.1. K-truss structure

The purpose of the community detection algorithm is to find a compact and dense subgraph structure. Luce and Perry (1949) proposed the concept of the clique structure, which meets the goal of community detection. The clique structure requires that any node in its subgraph is connected with all other nodes. However, finding clique structures in complex networks is a complex task, so Cohen (2008) proposed the concept of the K-truss structure. Unlike clique structures, K-truss structures only require that each edge in the subgraph belongs to at least (K-2) triangles. Additionally, the K-truss structure is more loosely defined than the clique structure (Cazals & Karande, 2008).

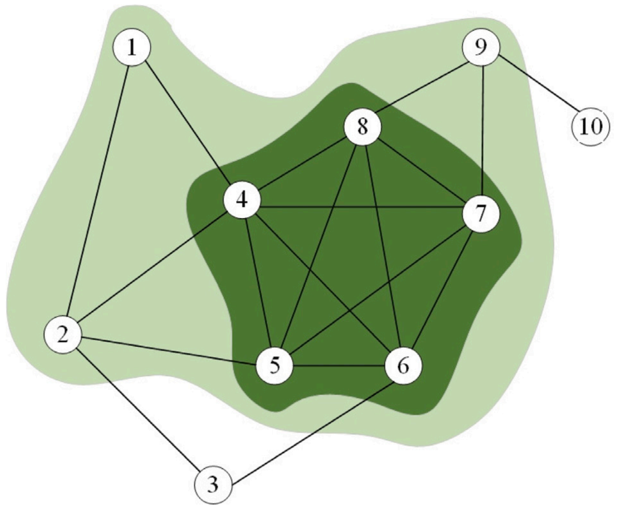

Fig. 1. Small network topology.

Support: In an undirected network $G = (V, E)$ and its subgraph G', for any edge $e \in G'$ in network G, $sup(e, G')$ is used to represent the support of edge e in subgraph G', that is, the number of triangles containing edge e in subgraph G', which is abbreviated as $sup(e)$.

K-truss structure: In an undirected network $G = (V, E)$, define subgraph $G_K(K \geq 2)$ as the K-truss structure in network G, and the support of any edge e in subgraph G_K is greater than or equal to $(K-2)$, which is recorded as $\forall e \in G_K, sup(e, G_K) \geq K - 2$.

Fig. 1 shows a small network topology. According to the definition of the K-truss structure, the network itself is a 2-truss structure. The subgraph represented by the light part in the network is a 3-truss structure, that is, each edge in the subgraph belongs to at least one triangle, and the subgraph represented by the dark part in the network is a 4-truss structure and a 5-truss structure, that is, each edge in the subgraph belongs to at least two or three triangles. It can be seen that the K-truss structure is a hierarchical and progressive subgraph structure. Different values of K characterize the network core structures with different densities. In this paper the K-truss structure is comprehensively used to find the core structure of the network to achieve a better community division effect.

3.1.2. Variational autoencoder

The variational autoencoder (VAE) (Kingma & Welling, 2013) is an unsupervised neural network model. It belongs to an "encoding-decoding" model, but its function and application are different from ordinary autoencoder. The generation model of variational autoencoder is alike to the generative adversarial networks (GANs) (Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, & Bengio, 2014). The variational autoencoder learns the hidden representation of the data through the encoding part, and decodes the hidden representation through the decoding part to reconstruct the input data.

In recent years, variational autoencoder have also been used in community detection (Chen, Chen, Zhang, Zheng, & Zou, 2019; Choong et al., 2020), fault detection (Bi & Zhao, 2021), anomaly detection (You et al., 2022) and other application fields. Based on real-world conditions, the variational autoencoder model is used in this study to feature extraction and dimension reduction to achieve a better clustering effect.

3.1.3. K-means algorithm

The K-means algorithm (Bahmani et al., 2012) is an iterative solution clustering algorithm, the steps of which are, pre-dividing all sample points into K groups, followed by randomly selecting K of them as the initial cluster centers, then calculating the distance between each sample point and each cluster center, and assigning each sample point to the cluster center closest to it. The clustering centers and the sample points assigned to them represent a cluster. Whenever a sample point is assigned, the cluster centers of the clusters are recalculated based on the existing sample points in the clusters. Finally the process is repeated until no more sample points are reassigned to different clusters. In this paper, sample points are equal to the number of nodes in the network, and we will use the K-means algorithm for clustering low-dimensional feature matrix.

3.2. CSEA algorithm

The CSEA algorithm is mainly divided into three steps: First, the process of searching for the core structure of the network using the K-truss algorithm is explained. Second, combined with the similarity matrix, the construction process of the variational autoencoder model is explained. Final, the K-means algorithm is used to cluster the low-dimensional feature matrix. The complete algorithm framework will be shown in Section 3.2.3.

3.2.1. The K-truss algorithm

Traditional community detection algorithms are mostly used to describe the relationship between network nodes based on the adjacency matrix, but there is a core structure in the real network. This core structure is shown as a compact and dense connection between some nodes in the network. Using only the adjacency matrix cannot highlight the core structure in the network, and the relationship representation between many key nodes is lost, which affects the accuracy of community division. Therefore, to combine the network core structure with the adjacency matrix, we first utilize the K-truss algorithm to find the core structure in real networks. We combine the local information with the core structure information to gain a new similarity matrix, which is used to describe the network structure information and improves the accuracy of community division.

Based on real networks, the core structure information is found through the K-truss algorithm. A detailed description of the K-truss algorithm is shown in Algorithm 1.

The input of Algorithm 1 is the original network G, and the output is the K-truss structure G_K. In steps 1–2, the parameters are initialized. We initialize the value of parameter K and create a priority queue pq. In steps 3–5, the process of calculating the support is represented. The support of each edge e in network G is calculated according to the set of adjacent nodes of edge e. In steps 7–15, the process of iteratively deleting the edges in the network is provided. When the queue is not empty, the support of the current edge e is obtained by reading the head element of the queue. Next, the edge with the lowest support is iteratively deleted, the current support is recalculated, and then the queue elements are sorted in ascending order according to the current support. In steps 16–17, the parameter updating process is shown. When there is an edge in the network G, the parameter K is updated and the iterative process continues. Finally, the output G_K is obtained.

Take Fig. 1 as an example to illustrate the process of iteratively deleting edges in the network. The network itself is a 2-truss structure. When $K = 2$, the edges with support $sup(e) \le 0$ in the network are deleted, that is, the edges $(2,3),(3,6),(9,10)$ are deleted, and the light colored 3-truss structure is obtained. When $K = 3$, the edges with support $sup(e) \le 1$ in the network are deleted, that is, the edges $(1,2),(1,4),(2,4),(2,5),(8,9),(7,9)$ are deleted (in the iteration process, when the edge $(1,2) or (1,4) or (2,5)$ is deleted, the support of the edge $(2,4)$ becomes $sup(e_{24}) = 1$, and is deleted immediately), and the dark colored 4-truss structure is obtained. When $K = 4$, the edges with support $sup(e) \le 2$ in the network are deleted, no edges are deleted, and the dark colored 5-truss structure is obtained.

Algorithm 1 K-truss Algorithm
Input: Network $G = (V, E)$
Output: K-truss structure G_K

```
1: $K = 2$
2: Initialize priority queue $pq$
3: for each $e_{ij} = (i, j) \in E$ do
4:     Calculate the adjacent node set of nodes $i$ and $j$ respectively, recorded as $S_i, S_j$
5:     $sup(e) = |S_i \cap S_j|$
6: $e_{ij}$ is sorted in ascending order according to the size of $sup(e)$ and stored into the queue $pq$
7: while $pq.isnotempty()$ and $pq.top() \le K - 2$ do
8:     for each $\omega \in (S_i \cap S_j)$ do
9:         $sup(\omega, i) = sup(\omega, i) - 1$
10:        $sup(j, \omega) = sup(j, \omega) - 1$
11:        $pq.push(e_{\omega i})$
12:        $pq.push(e_{j\omega})$
13:        The elements in the queue pq are sorted in ascending order according to the size of $sup(e)$
14:    $G_K = G_K \cup \{e\}$
15:    $pq.pop(e)$
16: $K = K + 1$
17: Goto step 8
18: Return $G_K$
```

To effectively combine the network core structure with the variational autoencoder, first, the network G is represented in a matrix to obtain the local information A, Then, the support of the edge corresponding to each K-truss structure in G_k obtained by Algorithm 1 is combined with the local information A through similarity mapping, and finally the similarity matrix $X = [x_{ij}]_{n\times n} \in R^{n\times n}$ is obtained.

Similarity: In an undirected network $G = (V, E)$, local information $A = [a_{ij}]_{n\times n}$, and the similarity of edge e_{ij} is defined as:

$$Sim(e_{ij}) = \sum_{i,j=0}^{n} \frac{a_{ij}(sup(e_{ij}) + 1)}{sup_{max}(e_{ij}) + 1} \tag{1}$$

where $sup(e_{ij})$ is the support of edge e_{ij} in network G and $sup_{max}(e_{ij})$ is the maximum support (derived from Algorithm 1).

Similarity matrix: In an undirected network $G = (V, E)$, with local information $A = [a_{ij}]_{n\times n}$, a new matrix $X = [x_{ij}]_{n\times n}$ can be obtained by calculating the similarity of nodes i and j corresponding to each edge e_{ij} according to Eq. (1), where $x_{ij} = Sim(e_{ij})$. Define matrix X as the similarity matrix.

According to Eq. (1), the elements in the similarity matrix X are mapped into the interval $(0, 1)$ by taking the support of each edge e as the "weight" in the weighted average to process the local information A. This is a kind of normalization method. At this time, $x_{ij} \in (0, 1)$. The similarity matrix X retains the feature information, that is, X contains the K-truss core structure information, which reflects the core structure in the real network. Compared with the local information A, the similarity matrix is used to better reflect the feature structure of the community and improve the accuracy of community division.

3.2.2. Variational autoencoder model

Due to the large-scale amount of the real networks, we use a variational autoencoder for feature extraction and dimension reduction.

The variational autoencoder learns the parameters of continuous probability distribution in the hidden space for random sampling and interpolation, and extracts 2 n-dimensional vectors, i.e., mean and standard deviation, through the encoding part, and subsequently samples them to obtain the corresponding random variables, and the n-dimensional sampling results formed after n times of sampling is used as the output of the encoding part, then, fed into the decoding part to complete reconstruction of data information. The variational autoencoder iteratively updates the parameters by back propagation algorithm, making the data output as equal as possible to the data input by reconstructing the error. The advantage of variational autoencoder is learning the parameters of continuous probability distribution in the hidden space, adding the process of minimizing KL divergence in the total loss function, increasing the fitting ability of the neural network, and better dimension reduction and feature extraction.

The variational autoencoder corresponds to the "encoding-decoding" model, where the part used for the generation of the hidden variables is called the encoding part and the part used for reconstructing the data input is called the decoding part. The learning of the hidden variables by the variational autoencoder results in an additional loss that provides a stochastic gradient variational Bayesian estimator for the training model.

Encoding: Define the posterior distribution $f_\phi(\cdot) = q_\phi(z|x)$ as the approximate inference process for the hidden variable Z.

Decoding: Define the conditional distribution $g_\theta(\cdot) = p_\theta(y|z)$ as the generating process of the generating variable Y.

Where X is an n-dimensional random vector based on the graph G representation i.e., the similarity matrix processed by the K-truss algorithm, $x_i \in R^{n\times 1}$ is the vector corresponding to the ith node of matrix X, Z is the hidden variable generated by the encoding part, $z_i \in R^{d\times 1}$ is the vector corresponding to the ith node of matrix Z, and d is the number of neurons in the hidden layer. Y is the generating variable generated by the decoding part that is as equal as possible to matrix X, $y_i \in R^{d\times 1}$ is the vector corresponding to the ith node of matrix Y, ϕ indicating the training parameters of the constrained encoding part, θ and indicating the training parameters of the constrained decoding part.

The basic equations for the encoding part and decoding part are shown in Eqs. (2) and (3), where $W_1 \in R^{d\times n}, b_1 \in R^{d\times 1}$ are the weights and biases in the encoding part, $W_2 \in R^{n\times d}, b_2 \in R^{1\times d}$ are the weights and biases in the decoding part, $\phi = \{W_1, b_1\}, \theta = \{W_2, b_2\}$, σ is a nonlinear activation function, such as Sigmoid function.

$$z_i = \sigma_f(W_1 x_i + b_1) \tag{2}$$

$$y_i = \sigma_g(W_2 z_i + b_2) \tag{3}$$

To make the generated data as equal as possible to the original input data and to reduce the loss, the loss function is defined as:

$$L(x_i, y_i) = -\frac{\sum_{i=1}^{n}[x_i log y_i + (1-x_i)log(1-y_i)]}{n} \tag{4}$$

Where n is the dimension of the matrix X, i.e., the number of nodes in the network.

Besides, KL divergence (Kullback & Leibler, 1951) is usually added to the loss function in a variational autoencoder, and the KL divergence serves to describe the divergence between two probability distributions.

As shown in Fig. 2, the hidden space variable Z is obtained by sampling from the Gaussian distribution determined by the mean vectors μ and σ^2, which leads to noise. The significance of minimizing the KL divergence is to optimize the probability distribution parameters μ and σ^2 so that they are as close as possible to the objective distribution and to ensure that the model has some generative power.

Define the KL divergence of the variational autoencoder model as:

$$\begin{aligned} KL(N(\mu,\sigma^2) \parallel N(0,1)) &= KL(q(z|x) \parallel p(z)) \\ &= L_{\mu,\sigma^2} = L_\mu + L_{\sigma^2} \\ &= \frac{1}{2}\sum_{i=1}^{m}(\mu_{x_i}^2 + \sigma_{x_i}^2 - log\sigma_{x_i}^2 - 1) \end{aligned} \tag{5}$$

Where, $L_\mu = \frac{1}{2}\sum_{i=1}^{m}\mu_{x_i}^2, L_{\sigma^2} = \frac{1}{2}(\sigma_{x_i}^2 - log\sigma_{x_i}^2 - 1)$, m is the dimension of the hidden variable.

Therefore, the final loss function of the variational autoencoder model constructed in this paper is:

$$Loss(\{\theta,\phi\}) = \min_{\{\theta,\phi\}}(\alpha\sum_{i=1}^{n}L(x_i,y_i) + \beta\sum_{j=1}^{m}KL(N(\mu,\sigma^2) \parallel N(0,1))) \tag{6}$$

where α, β are the weighting factors. The training parameters $\{\theta,\phi\}$ are continuously updated through iteration until the algorithm converges.

The feedforward propagation process of the variational autoencoder model constructed in this paper is shown in Algorithm 2.

Algorithm 2 Feedforward propagation process of variational autoencoder model

```
Input:
Similarity matrix $X$
Number of nodes in each layer of neural network $D = \{d_1, d_2, \cdots, d_s\}$
Output:
Matrix after one training period

1: Calculate the number of layers of the neural network $s$ (Dimensionality of $D$)
2: Create an input layer based on the dimension of $d_1$
3: for each $d_i \in D, (1 < i < \lfloor s/2 \rfloor)$ do
4:     Set the activation function $\sigma_f$
5:     Create hidden layers based on the dimension of $d_i$
6:     Calculate a feedforward propagation process according to Eq. (2)
7: A reparameterization layer is established according to the dimension $m$ of hidden layer $d_{\lceil s/2 \rceil}$
8: A parameter $\varepsilon$ is sampled from Gaussian distribution $N\{0,1\}$
9: The vector representation of $Z$ is sampled from the distribution $N(\mu,\sigma^2)$
10: Calculate the result of reparameterization layer and update $Z = \mu + \varepsilon \times \sigma$
11: for each $d_i \in D, (\lceil s/2 \rceil + 1 < i < s)$ do
12:     Set the activation function $\sigma_g$
13:     Create hidden layers based on the dimension of $d_i$
14:     Calculate a feedforward propagation process according to Eq. (3)
15: Return Matrix after one training period
```

Algorithm 2 illustrates the feedforward propagation process of variational autoencoder model. The input of algorithm 2 is the similarity matrix X and the number of nodes in each layer of neural network D, the output is a matrix after one training period. Among them, steps 2–6 describe the feedforward propagation process of the encoding part, steps 7–10 describe the calculation process of the reparameterization parameter layer, which is equivalent to adding noise and making the whole model trainable, steps 11–14 describe the feedforward propagation process of the decoding part, and take the reparameterization parameter layer result Z as the input of the decoding part.

3.2.3. Framework

The basic framework of the CSEA algorithm model is introduced in this section. As shown in Fig. 2, the framework is divided into 3 steps: data preprocessing based on the K-truss algorithm, which is combined with the K-truss structure to convert local information into a similarity matrix (see Section 3.1); based on the deep learning link of the variational autoencoder, the similarity matrix is transformed into a low-dimensional feature matrix (see Section 3.2); and community information is obtained by clustering the low-dimensional feature matrix based on the K-means algorithm (see this Subsection).

Algorithm 3 illustrates the basic steps of the CSEA algorithm. The main inputs of the CSEA algorithm are the original network graph G and the parameter p, the output is the community information C'_p. Steps 1–3 are the data preprocessing part (see Section 3.1), steps 4–15 are the

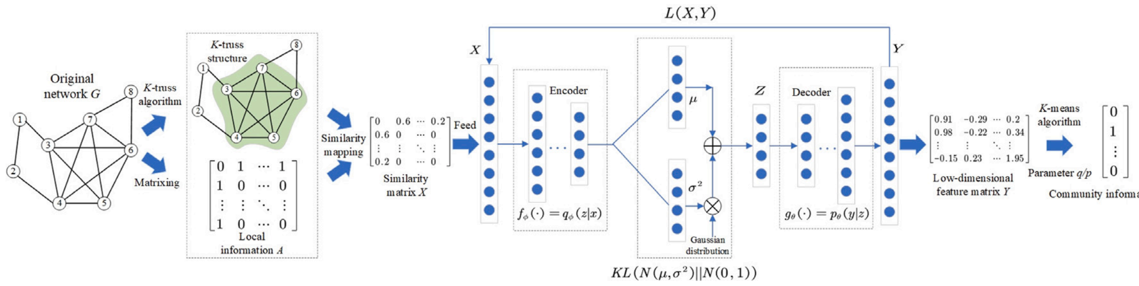

Fig. 2. Framework of the CSEA algorithm model.

Algorithm 3 CSEA Algorithm Model

```
Input:
Original network $G = (V, E)$
the number of different kinds of community labels $p$
Weighting factors $\alpha, \beta$
Number of nodes in each layer of neural network $D = \{d_1, d_2, \cdots, d_s\}$
Number of iterations $IRS$
Output:
Community information $C'_p$

1: Compute the adjacency matrix of network $G$ to obtain the local information $A$
2: Use Algorithm 1 to find the K-truss structure of the graph $G$ and obtain the subgraph $G_k$
3: Use Eq. (1) to obtain the similarity matrix $X$
4: Initialize the variational autoencoder, initialize the set of parameters $\{\theta, \phi\}$
5: $i = 1$
6: while $i < IRS$ do
7:     Input similarity matrix $X$
8:     Calculate the number of layers of the neural network $s$ (Dimensionality of $D$)
9:     Calculate the feedforward propagation process of the encoding part according to steps 2-6 of Algorithm 2
10:    The hidden layer representation $Z$ is obtained by calculating the reparameterization layer according to steps 7-10 of Algorithm 2
11:    Calculate KL divergence according to Eq. (5)
12:    Calculate the feedforward propagation process of the decoding part according to steps 11-14 of Algorithm 2
13:    Calculate the total loss function according to Eq. (6)
14:    Update parameters $\{\theta, \phi\}$
15:    $i = i + 1$
16: Obtain the low-dimensional feature matrix $Y$
17: Clustering matrix $Y$ using K-means algorithm
18: Return Community information $C'_p$
```

deep learning part (see Section 3.2), and steps 17–18 are the clustering part. The matrix $Y = [y_{ij}]_{n\times m}$ is generated from the similarity matrix X trained by the variational autoencoder model and clustered by the K-means algorithm to obtain the community information $C'_p = [c'_i]_{n\times 1} \in R^{n\times 1}$.

The time complexity of the CSEA algorithm is divided into three parts: In the first step, $deg(i)$ is the degree of node i, the process of calculating the support by the K-truss algorithm (steps 3–5 in Algorithm 1) can be accelerated and optimized using the memory triangular computation algorithm (Latapy, 2008). However, the process of iteratively deleting edges in the network (steps 7–15 in Algorithm 1) needs to update the support, at which time processing each edge still takes $O(deg(i) + deg(j))$, so the time complexity is $O(\sum_{i\in V}(deg(i))^2)$ (Wang & Cheng, 2012; Zheng et al., 2017). In the second step, the time complexity of the deep learning part is $O(s \times IRS)$ (Xie et al., 2019; Xu et al., 2020), the essence is to calculate the time complexity of feedforward and backward propagation during the training of the variational autoencoder neural network. s is the number of the neural network layer, and IRS is the number of iterations of the neural network. The third step, the complexity of K-means algorithm is $O(IRSK \times p \times n)$, where $IRSK$ is the number of iterations of K-means algorithm, p is the number of clusters, and n is the number of sample points(the number of nodes in the network) of the K-means. Therefore, the total time complexity of the CSEA algorithm model is the sum of three steps: $O(\sum_{i\in V}(deg(i))^2 + s \times IRS + IRSK \times p \times n)$.

4. Experiment and analysis

In this section, the proposed method is evaluated and tested based on different real networks. First, the related datasets, evaluation metrics and comparison algorithms are introduced. Second, 4 evaluation metrics are used to evaluate the community detection performance of the CSEA algorithm with other 17 community detection algorithms on 19 real networks. Final, community visualization and the comparison of algorithm efficiency are realized.

4.1. Experimental settings

The experiment is completed using the Windows 10 operating system. We used the Python languages (Python 3.8.2, TensorFlow 2.8.0) and C++ language. The hardware environment is Intel Core i7-9700k CPU, with 48 GB of memory and an NVIDIA GeForce RTX 2080 GPU.

As shown in Table 1, 11 networks with real community labels and 8 networks without real community labels are used in this experiment. The parameter "/" indicates that the number of communities in this network is unknown. Among these networks, YouTube, DBLP, LiveJournal and Orkut are subnetworks formed by extracting nodes from the complete networks. The first two extract 11 and 12 communities respectively. The nodes and edges of connecting nodes in these communities are used to form a new subnetwork. The latter two are used in the community detection without real community labels. According to the scale of the complete networks, the number of nodes and edges extracted are shown in Table 1, where n represents the number of nodes in the network, m represents the number of edges in the network, q represents the number of different kinds of real community labels, and $\overline{q}$ represents the average node degree. The network datasets are available at: http://wwwpersonal.umich.edu/~mejn/netdata/, https://linqs.soe.ucsc.edu/data and http://snap.stanford.edu/data/index.html.

Table 2 shows the size of the neural network on different networks. The size of the input layers is the number of nodes n owned by different networks. Due to the influence of the number of neural network layers and the number of nodes in each layer on the experimental results of the CSEA algorithm. Therefore, we give some basic parameter settings, and in most cases, by using this parameter settings, the best community division can often be obtained. However, this parameter settings does

Table 1
Datasets statistics.

Networks	n	m	q	$\overline{q}$
Strike (Khorasgani, Chen, & Zaiane, 2010)	24	38	3	3.17
Karate (Zachary, 1977)	34	78	2	4,59
Dolphins (Lusseau, Schneider, Boisseau, Haase, Slooten, & Dawson, 2003)	62	159	2	5.13
Polbooks (Newman, 2006b)	105	441	3	8.40
Football_12 (Khorasgani et al., 2010)	180	788	12	8.74
Email_Eu (Yin, Benson, Leskovec, & Gleich, 2017)	1005	25571	42	33.25
Polblogs (Girvan & Newman, 2002)	1490	16717	2	22.44
Cora (Sen, Namata, Bilgic, Getoor, Galligher, & Eliassi-Rad, 2008)	2708	5278	7	3.90
Citeseer (Sen et al., 2008)	3312	4536	6	2.74
Youtube (Yang & Leskovec, 2015)	7641	35856	11	9.39
DBLP (Yang & Leskovec, 2015)	13040	44318	12	6.80
Lesmis (Knuth, 1993)	77	254	/	6.60
Jazz (Gleiser & Danon, 2003)	198	2742	/	27.70
Neural (Watts & Strogatz, 1998)	297	2148	/	14.46
Metabolic (Duch & Arenas, 2005)	453	2025	/	9.01
Email_URV (Guimera, Danon, Diaz-Guilera, Giralt, & Arenas, 2003)	1133	5451	/	9.62
Livejournal (Yang & Leskovec, 2015)	6386	90599	/	28.45
Orkut (Yang & Leskovec, 2015)	8929	138690	/	31.07
PGP (Boguná, Pastor-Satorras, Díaz-Guilera, & Arenas, 2004)	10680	24316	/	4.55

Table 2
Layer settings of the variational autoencoder network.

Networks	Layer settings	Networks	Layer settings
Strike	24–18	DBLP	4096-2048-1024-512-256
Karate	24–18	Lesmis	36-18-16
Dolphins	32-24-12	Jazz	80-40-36
Polbooks	32–24	Neural	96-64-72
Football_12	36–32	Metabolic	80-60-60
Email_Eu	64-48-32-28-12	Email_URV	128-96-64-72-72
Polblogs	256-128-64-128-90	Livejournal	96-72-64-48-32
Cora	128-64-256	Orkut	2048-1024-512-1024-512
Citeseer	72-64-48-32	PGP	128-256-64-96
Youtube	1024-512-256-128-96		

not optimize all 4 evaluation metrics at the same time in some networks, so it is necessary to adjust the corresponding parameter settings to optimize one or several metrics. In addition, in the variational autoencoder model, we use the Adamax optimizer, and set the learning rate $lr = 0.014$ and the weighting factors $\alpha = \beta = 0.5$.

4.2. Evaluation metrics

For the community information C'_p and real community information C_q, when the real community labels are known, set $p = q$. In this paper, 4 common community evaluation metrics, MNI, F_{same}, ACC and Q, are used to measure the accuracy of community division.

The NMI (Danon, Diaz-Guilera, Duch, & Arenas, 2005) is a widely used similarity measure, which represents the similarity between real community division and community division obtained by community detection algorithm. The equation is as follows:

$$NMI(C, C') = -\frac{2\sum_{i=1}^{q}\sum_{j=1}^{p} n_{ij} log(\frac{n_{ij}n}{n_i n_j})}{\sum_{i=1}^{q} n_i log(\frac{n_i}{n}) + \sum_{j=1}^{p} n_j log(\frac{n_j}{n})} \tag{7}$$

where n is the number of nodes in network G, p is the number of different kinds of community labels, and q is the number of different kinds of real community labels. n_{ij} indicates the number of nodes that appear in both the ith detected community C'_i and the jth real community C_j. n_i and n_j are the total number of nodes in C'_i and C_j, respectively. The value range of NMI is 0 to 1.

The F_{same} (Raghavan et al., 2007) is used to measure the degree of cross similarity between the community information and the real community information. The equation is as follows:

$$F_{same}(C, C') = \frac{\sum_{i=1}^{p} max|c_i \cap c'_j| + \sum_{j=1}^{q} max|c_i \cap c'_j|}{2n} \tag{8}$$

where n is the number of nodes in network G, p is the number of different kinds of community labels, and q is the number of different kinds of real community labels. The value range of F_{same} is 0 to 1. Different from other evaluation metrics, F_{same} requires the parameter q as the input, which shows that F_{same} can evaluate the community detection algorithm with a number of different kinds of real community labels as the input parameter well.

The accuracy ACC (Qin, Jin, Lei, Gabrys, & Musial-Gabrys, 2018) is used to obtain the predicted cluster allocation from the real community information C_i and community information C'_j, and find the best mapping between them. The equation is as follows:

$$ACC(C, C') = \frac{\sum_{i=1}^{n} \delta(c_i, m(c'_i))}{n} \tag{9}$$

where n is the number of nodes in network G. $\delta(x, y) = 1$ if and only if $x = y$, otherwise 0. $m(c'_i)$ is the Hungarian mapping function (Kuhn, 1955), which maps the community information label c'_i to its corresponding real value. The value range of ACC is 0 to 1.

The modularity Q (Clauset et al., 2004; Newman & Girvan, 2004) measures the community detection algorithm from the perspective of network topology. The equation is as follows:

$$Q = \frac{1}{2m}\sum_{ij}(a_{ij} - \frac{deg(i)deg(j)}{2m})\delta(c'_i, c'_j) \tag{10}$$

where m is the number of nodes in network G, $deg(i)$ is the degree of node i, and a_{ij} is an element in the local information. If connecting edge existing between node i and node j, $a_{ij} = 1$; otherwise it is 0, $\delta(x, y) = 1$ if and only if $x = y$; otherwise it is 0. The value range of is 0 to 1. Moreover, Q does not need to take the real community information label as the input parameter, which indicates that Q can be a measure of the networks without real community labels. Meanwhile, in this experiment, the performance of the community detection algorithm needs to be evaluated by combining 4 evaluation metrics because some algorithms do not take the number of different kinds of real community labels as input parameters, thus there is a large variation in the value of Q.

4.3. Comparison algorithm

In order to compare the performance and efficiency of CSEA algorithm, total 18 different kinds of algorithms were compared. In addition, we also compare some metrics of MNDP and DNR on some networks based on experimental data from the literature. The description of these networks is as follows:

FN (Newman, 2004): A hierarchical clustering algorithm on the basis of greedy strategy.

SC (Amini et al., 2013): A classical clustering algorithm built on spectral graph theory, which constructs feature vectors by calculating the Laplacian matrix.

KL (Kernighan & Lin, 1970): A tentative greedy algorithm for network node division.

NMF (Gillis & Glineur, 2012): A Nonlinear dimension reduction that is achieved by factorizing the original matrix and clustered using the K-means algorithm.

MDNP (Jin et al., 2015): An improved nonnegative matrix factorization method that preserves the node degree while promoting a modular null model.

DNR (Yang et al., 2016): A community detection algorithm on the basis of modularity and deep learning models. Among these models, DNRL2 is a DNR method that uses the L2 paradigm and DNRCE is a DNR method that uses the cross-entropy distance.

Deepwalk (Perozzi et al., 2014): A network embedding method based on random walks, where a low-dimensional representation is obtained and clustered using the K-means algorithm.

Node2vec (Grover & Leskovec, 2016): Based upon the Deepwalk algorithm, node2vec combines both depth-first and breadth-first walks to obtain a low-dimensional representation and clustered using the K-means algorithm.

PCA (Shlens, 2014): A dimension reduction method that can be used to extract the main feature information of the data, where a low-dimensional representation is obtained and clustered using the K-means algorithm.

DSACD (Fei et al., 2020): A dimension reduction and feature extraction network structure that uses a similarity matrix and a deep sparse autoencoder, where a low-dimensional representation is obtained and clustered using the K-means algorithm.

GN (Newman, 2013): A classical divide hierarchical clustering algorithm.

LE (Newman, 2006a): A method for maximizing modularity using the eigenvalue factorization of the modularity matrix.

LPA (Raghavan et al., 2007): A graph-based semi-supervised learning algorithm for building complete graph models using the idea of label propagation.

CNM (Clauset et al., 2004): An agglomerate hierarchical clustering algorithm based on the idea of a greedy algorithm, which uses a "heap" data structure to compute and update the modularity of the network.

FUA (Blondel et al., 2008): An extensive modularity-based community detection algorithm whose optimization objective is to maximize the modularity of the whole community network.

Infomap (Rosvall & Bergstrom, 2008): A community detection algorithm based on random walks that combines information encoding and community detection to find the best community division by minimizing the encoding length.

Edmot (Li, Huang, Wang, & Lai, 2019): A new advanced Motif-aware edge enhancement method for community detection.

CDME (Sun, Sun, Chang, Wang, Yan, Pan, & Li, 2020): A new community detection method based on the Matthew effect, which designs a new dynamics model to simulate the Matthew effect, treats the network as a social system, and studies the dynamics over time.

4.4. Algorithm model performance evaluation

We conduct experiments based on whether the network has real community labels and use 4 evaluation metrics and 18 community detection algorithms.

Table 3
Comparison of 9 different methods based on their number of clusters.

Networks	CSEA(q)	GN	KL	LE	LPA	CNM	FUC	Infomap	Edmot	CDME
Strike	3	3	2	3	7	4	4	4	4	3
Karate	2	2	2	2	3	3	4	3	4	2
Dolphins	2	2	2	2	6	4	5	5	5	4
Polbooks	3	3	2	3	4	4	5	6	5	3
Football_12	12	12	2	7	9	12	9	13	8	11
Email_Eu	42	61	2	26	38	61	27	328	27	21
Polblogs	2	2	2	268	278	277	276	306	/	4
Cora	7	2	2	78	481	105	105	319	104	219
Citeseer	6	2	2	438	872	487	470	671	/	543
Youtube	11	/	2	7	125	44	18	332	/	17
DBLP	12	/	2	23	1680	146	71	998	/	774

4.4.1. Real networks with ground-truth

In this subsection, the parameter p can also be considered an evaluation metric because there are some comparison algorithms that do not need to use the parameter q as an input parameter, which results in the number of clusters obtained by these algorithms that are not consistent with the parameter q.

The number of clusters obtained by different community detection algorithms is shown in Table 3, and other algorithms not appearing in Table 3 are the same as the CSEA algorithm, which requires the parameter q as the input, i.e., $p = q$.

In the 11 real networks, the experimental results are shown in Tables 4–7, where (a) indicates that the CSEA algorithm is compared with the algorithm with the parameter q input, and (b) indicates that the CSEA algorithm is compared with the algorithm without the parameter q input. With respect to the FN, SC, GN, and DSACD algorithms, "/" represents the algorithm running over $18h$ or being unable to handle the memory consumption. For the MNDP, DNRL2, and DNRCE algorithms, "/" represents data cases where the relevant network is not given in the literature. For the Edmot algorithm, "/" means that the experimental results on this network are not available in the experimental environment we built. In Table 5, the DNR algorithm is not compared because the data for this metric are not given in the relevant literature. Similarly, in Tables 6–7, the MNDP and DNR algorithms are not compared because the data using this metric are not given in the relevant literature. In addition, in Table 7, since F_{same} requires q as the input, only the relevant eligible algorithms can be compared.

As shown in Table 4, using the NMI metric compared with other algorithms, the best results on Strike, Karate, Dolphins, Polbooks, Football_12, Emial_Eu and Polblogs are achieved using the CSEA algorithm, with results of 1, 1, 1, 0.609, 0.989, 0.668 and 0.573, respectively, except for Cora, Citeseer, YouTube and DBLP. For the algorithms with q input, the best results with 0.668, 0.385, 0.446, 0.611 and 0.339 on the Emai_Eu, Cora, Citeseer, YouTube and DBLP, respectively, are achieved using node2vec. The second best results on the YouTube and DBLP are achieved by using the CSEA algorithm. For the algorithms without q input, CDME achieves the best result of 0.674 on Polblogs, CNM achieves the best result of 0.466 on Cora, while FUA achieves the best result of 0.699 and 0.596 on the Youtube and DBLP, respectively.

As shown in Table 5, using the ACC metric, compared to the other algorithms, the best results on Strike, Karate, Dolphins, Polbooks, Football_12, Emial_Eu, YouTube, with results of 1, 1, 1, 0.848, 0.994, 0.525 and 0.557, respectively, are achieved using the CSEA algorithm. For the algorithms with q input, the best result on Polblogs(0.947) is achieved using MNDP. And the second best result with a result of 0.902 is achieved using the CSEA algorithm. The best results on Cora, Citeseer and DBLP are achieved using node2vec. The second best results on Citeseer and DBLP are achieved using the CSEA algorithm. For the algorithms without q input, CDME achieves the best result of 0.623 on the Polblogs, while on all other networks, the best results for the algorithms without parameter q input are lower (or equal) than the best results for the algorithms with parameter q input.

Table 4
Comparison of 18 methods in terms of NMI on 11 real networks.

(a)

Networks	CSEA	FN	SC	NMF	MNDP	DNRL2	DNRCE	Deppwalk	Node2vec	PCA	DSACD
Strike	**1.000**	0.866	0.649	0.865	/	/	/	0.866	0.866	0.736	**1.000**
Karate	**1.000**	0.837	0.836	**1.000**	**1.000**	**1.000**	**1.000**	0.646	0.836	**1.000**	**1.000**
Dolphins	**1.000**	0.814	0.408	0.284	0.889	0.889	0.818	0.699	0.889	0.241	0.777
Polbooks	**0.609**	0.534	0.529	0.349	0.530	0.552	0.582	0.579	0.563	0.338	0.351
Football_12	**0.989**	0.530	0.626	0.958	/	/	/	0.677	0.665	0.946	0.898
Email_Eu	**0.668**	0.494	0.563	0.534	/	/	/	0.457	**0.668**	0.473	0.415
Polblogs	**0.573**	0.326	0.266	0.130	0.711	0.389	0.517	0.417	0.446	0.129	0.014
Cora	0.264	0.374	0.094	0.168	0.340	/	/	0.298	**0.385**	0.140	0.064
Citeseer	0.109	0.109	0.103	0.071	0.340	/	/	0.080	**0.446**	0.080	0.134
Youtube	0.566	/	/	0.401	/	/	/	0.359	**0.611**	0.318	/
DBLP	0.203	/	/	0.051	/	/	/	0.116	**0.339**	0.035	/

(b)

Networks	CSEA	GN	KL	LE	LPA	CNM	FUA	Infomap	Edmot	CDME
Strike	**1.000**	**1.000**	0.655	**1.000**	0.651	0.770	0.866	0.884	0.770	**1.000**
Karate	**1.000**	0.836	0.837	**1.000**	0.445	0.692	0.687	0.699	0.587	**1.000**
Dolphins	**1.000**	0.889	0.136	0.753	0.527	0.573	0.593	0.586	0.593	0.579
Polbooks	**0.609**	0.575	0.569	0.528	0.534	0.531	0.560	0.493	0.556	0.575
Football_12	**0.989**	0.633	0.309	0.476	0.643	0.545	0.621	0.671	0.610	0.650
Email_Eu	**0.668**	0.106	0.116	0.393	0.180	0.494	0.556	0.600	0.566	0.238
Polblogs	0.573	0.116	0.463	0.115	0.385	0.378	0.376	0.331	/	**0.674**
Cora	0.264	0.158	0.052	0.120	0.409	**0.466**	0.429	0.413	0.448	0.418
Citeseer	0.109	0.299	0.006	0.247	**0.343**	0.331	0.227	0.332	/	0.328
Youtube	0.566	/	0.283	0.526	0.662	0.622	**0.699**	0.564	/	0.694
DBLP	0.203	/	0.023	0.227	0.490	0.458	**0.596**	0.505	/	0.519

Table 5
Comparison of 17 methods in terms of ACC on 11 real networks.

(a)

Networks	CSEA	FN	SC	NMF	MNDP	Deepwalk	Node2vec	PCA	DSACD
Strike	**1.000**	0.958	0.792	0.958	/	0.958	0.958	0.917	**1.000**
Karate	**1.000**	0.971	0.971	**1.000**	**1.000**	0.912	0.971	**1.000**	**1.000**
Dolphins	**1.000**	0.968	0.839	0.677	0.984	0.952	0.984	0.629	0.968
Polbooks	**0.848**	0.819	0.790	0.543	0.819	0.829	0.838	0.600	0.629
Football_12	**0.994**	0.383	0.650	0.911	/	0.650	0.650	0.928	0.911
Email_Eu	**0.525**	0.337	0.393	0.386	/	0.301	0.505	0.358	0.250
Polblogs	0.902	0.726	0.702	0.586	**0.947**	0.826	0.844	0.580	0.542
Cora	0.462	0.530	0.248	0.344	0.444	0.485	**0.532**	0.304	0.284
Citeseer	0.335	0.312	0.324	0.238	0.288	0.332	**0.426**	0.245	0.325
Youtube	**0.557**	/	/	0.443	/	0.421	0.474	0.359	/
DBLP	0.256	/	/	0.142	/	0.248	**0.433**	0.139	/

(b)

Networks	CSEA	GN	KL	LE	LPA	CNM	FUA	Infomap	Edmot	CDME
Strike	**1.000**	**1.000**	0.792	**1.000**	0.583	0.833	0.958	0.875	0.833	**1.000**
Karate	**1.000**	0.971	0.971	**1.000**	0.794	0.735	0.676	0.824	0.647	**1.000**
Dolphins	**1.000**	0.984	0.694	0.952	0.613	0.694	0.613	0.581	0.613	0.629
Polbooks	**0.848**	0.838	0.838	0.781	0.800	0.810	0.790	0.695	0.790	0.838
Football_12	**0.994**	0.533	0.261	0.400	0.544	0.428	0.500	0.656	0.467	0.572
Email_Eu	**0.525**	0.127	0.127	0.277	0.199	0.337	0.405	0.127	0.419	0.116
Polblogs	**0.902**	0.426	0.877	0.428	0.768	0.758	0.751	0.686	/	0.886
Cora	**0.462**	0.311	0.264	0.285	0.180	0.416	0.355	0.122	0.360	0.265
Citeseer	**0.335**	0.296	0.233	0.178	0.075	0.227	0.181	0.052	/	0.110
Youtube	0.557	/	0.256	0.231	0.542	0.469	0.429	0.507	/	**0.623**
DBLP	**0.256**	/	0.136	0.133	0.062	0.179	0.213	0.056	/	0.133

As seen from Table 6, using the Q metric, for the algorithms with parameter q input, the best results on the Strike, Dolphins and Polbooks with values of 0.548, 0.403 and 0.522, respectively, are achieved using the CSEA algorithm. The best results on Football_12, YouTube and DBLP are achieved using node2vec, while the best results on Karate, Emial_Eu, Polblogs, Cora and Citeseer are achieved using FN. Meanwhile, the second best results on Karate, YouTube, and DBLP are achieved using the CSEA algorithm. For the algorithms without parameter q input, the best results on Strike and Polbooks are achieved using Infomap, and the best results on the networks other than Strike, Polbooks and Citeseer are achieved using FUA. In addition, the best results on Polbooks and Cora are achieved using Edmot, the best result on Football_12 is achieved using GN, and the best results on Polblogs and Citeseer are achieved using CNM. At the same time, we can see that the best results in the algorithms without parameter q input are higher (or equal) than the best results in the algorithms with parameter q input on all the networks except Polblogs.

As can be seen from Table 7, using the F_{same} metric, the best results on all networks except Cora, Citeseer and YouTube, with values of 1, 1, 1, 0.857, 0.994, 0.606, 0.902 and 0.631, are achieved using the CSEA algorithm. The best results on Cora and Citeseer, with the values of 0.621 and 0.599, respectively, are achieved using NMF. The best result on DBLP with a value of 0.448, is achieved using node2vec. The visualization results of Table 7 are shown in Fig. 3.

Table 6
Comparison of 16 methods in terms of Q on 11 real networks.

(a)

Networks	CSEA	FN	SC	NMF	Deepwalk	Node2vec	PCA	DSACD
Strike	**0.548**	0.547	0.186	0.521	0.547	0.547	0.284	**0.548**
Karate	0.371	**0.372**	0.360	0.371	0.313	0.360	0.371	0.371
Dolphins	**0.403**	0.385	0.231	0.304	0.348	0.379	0.264	0.359
Polbooks	**0.522**	0.501	0.439	0.307	0.499	0.502	0.303	0.267
Football_12	0.517	0.535	0.237	0.483	0.552	**0.570**	0.476	0.464
Email_Eu	0.269	**0.327**	0.182	0.180	0.205	0.246	0.184	0.114
Polblogs	0.425	**0.427**	0.351	0.240	0.423	0.425	0.223	0.030
Cora	0.611	**0.758**	0.169	0.178	0.706	0.734	0.181	0.135
Citeseer	0.602	**0.783**	0.369	0.181	0.682	0.745	0.243	0.494
Youtube	0.472	/	/	0.220	0.380	**0.510**	0.184	/
DBLP	0.730	/	/	0.155	0.658	**0.755**	0.088	/

(b)

Networks	CSEA	GN	KL	LE	LPA	CNM	FUA	Infomap	Edmot	CDME
Strike	0.548	0.548	0.418	0.548	0.474	0.556	0.547	**0.562**	0.556	0.548
Karate	0.371	0.360	0.372	0.371	0.325	0.381	**0.420**	0.402	0.419	0.371
Dolphins	0.403	0.379	0.336	0.390	0.499	0.495	0.528	**0.529**	0.528	0.526
Polbooks	0.522	0.483	0.457	0.455	0.481	0.502	**0.527**	0.523	**0.527**	0.483
Football_12	0.517	**0.592**	0.349	0.500	0.574	0.535	**0.592**	0.575	0.585	0.571
Email_Eu	0.269	−0.016	0.201	0.341	0.070	0.347	**0.413**	0.060	0.392	−0.036
Polblogs	0.425	0.001	0.426	0.000	0.426	**0.427**	**0.427**	0.423	/	0.411
Cora	0.611	0.131	0.400	0.078	0.652	0.807	**0.815**	0.715	**0.815**	0.742
Citeseer	0.602	0.660	0.439	0.346	0.747	**0.871**	0.814	0.814	/	0.816
Youtube	0.472	/	0.411	0.615	0.635	0.627	**0.670**	0.615	/	0.621
DBLP	0.730	/	0.427	0.573	0.775	0.825	**0.921**	0.828	/	0.845

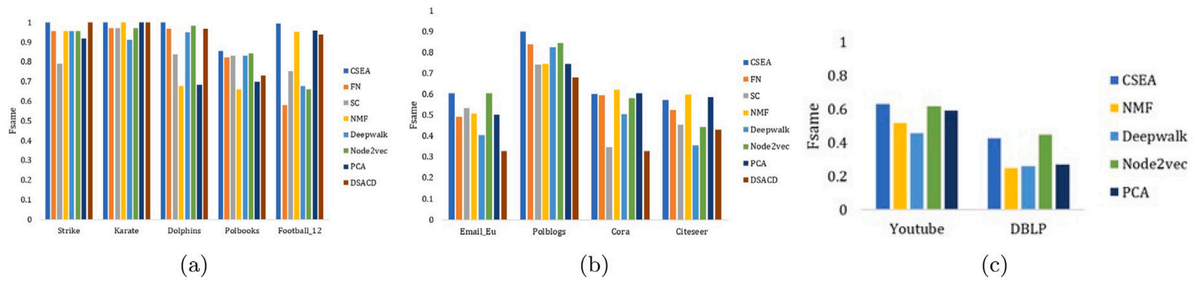

Fig. 3. Fsame results visualization.

Table 7
Comparison of 7 methods in terms of Fsame on 11 real networks.

Networks	CSEA	FN	SC	NMF	Deepwalk	Node2vec	PCA	DSACD
Strike	**1.000**	0.958	0.792	0.958	0.958	0.958	0.917	**1.000**
Karate	**1.000**	0.971	0.971	**1.000**	0.912	0.971	**1.000**	**1.000**
Dolphins	**1.000**	0.968	0.839	0.677	0.952	0.984	0.685	0.968
Polbooks	**0.857**	0.824	0.833	0.662	0.833	0.843	0.700	0.733
Football_12	**0.994**	0.581	0.753	0.953	0.677	0.661	0.961	0.939
Email_Eu	**0.606**	0.492	0.535	0.507	0.406	0.604	0.501	0.328
Polblogs	**0.902**	0.840	0.744	0.745	0.826	0.844	0.746	0.681
Cora	0.601	0.595	0.347	**0.621**	0.505	0.582	0.606	0.328
Citeseer	0.573	0.527	0.455	**0.599**	0.356	0.444	0.588	0.432
Youtube	**0.631**	/	/	0.517	0.458	0.617	0.593	/
DBLP	0.427	/	/	0.250	0.261	**0.448**	0.268	/

Based on Tables 4–7, we find that better results on 3 evaluation metrics, NMI, ACC and F_{same}, are achieved when using the CSEA algorithm compared with other algorithms, especially in small and medium-sized networks. However, the ideal results on Q are not achieved when using the CSEA algorithm. This is because all of these algorithms utilize the idea of maximizing modularity, as seen from Table 3. As a result, the parameter p obtained by these algorithms is greater than the parameter q. Taking Cora and Citeseer as examples, the parameter p for the LPA, CNM, FUA, and Infomap algorithms are 481, 105, 105, 319, and 872, 487, 470 and 671, respectively, while the parameter q for these two networks are 7 and 6, respectively. Meanwhile, according to the results from using these algorithms, the other 3 metrics are less effective when using the CSEA algorithm and other algorithms that have the parameter q input, indicating that these algorithms maximize Q while ignoring the division of the real community. Using the idea of maximizing modularity for community detection may lead to the real network being divided into many very small sets of communities, which also shows that evaluating algorithm performance based on Q alone is one-sided. Jin et al. (2020), Lu et al. (2020) also demonstrate this point.

The CSEA algorithm achieves better results on the 7 networks Strike, Karate, Dolphins, Polbooks, Football_12, Emial_Eu and Polblogs on the 3 metrics NMI, ACC and F_{same}. There are two main reasons for this. The first reason: since the number of clusters of the algorithm without parameter q input and the number of different kinds of real community labels q have a difference, the 2 metrics NMI and ACC evaluate the real and accurate degree of community detection algorithm division, and the community detection algorithm with the number of clusters closer to the parameter q is closer to the real community division, and thus can achieve better performance. The second reason is also the main reason: the above 7 networks with better results are small and medium-sized dense networks compared to the other networks. In dense networks, there are more edges connected between nodes than in sparse networks, which indicates that the average node degree $\overline{q}$ is larger in dense networks than in sparse networks. A large average node degree $\overline{q}$ implies that there are more edges connected between nodes in the network, a large degree of network aggregation, and correspondingly, fewer nodes of low degree. Therefore, in dense networks, it is easier to extract the high truss structure in the network using the K-truss algorithm (the value of K in the K-truss algorithm is higher), which can find the more core structure in the network, then perform feature extraction by the variational autoencoder model, and the extracted core structure information of the network is more obvious. Meanwhile, in small and medium-sized networks, the core structure of the network itself is more obvious due to the limitation of network topology with a small network, so a better community division can be obtained by using the CSEA algorithm. Therefore, the CSEA algorithm model can get a better community division in denser networks, especially in small and medium-sized dense networks. As for the algorithm with parameter q input, the same, NMI, ACC and F_{same} 3 metrics on the evaluation is the real and accurate degree of community detection algorithm, due to the second characteristic of the CSEA algorithm, which enables it to get a better performance.

Compared with other small and medium-sized networks, good performance on Cora, Citeseer and DBLP is not achieved when using the CSEA algorithm, meanwhile, better results on 2 metrics, ACC and F_{same}, on YouTube. On Cora, Citeseer and DBLP, according to Table 1, under the networks of 1000-level or 10000-level, the average node degrees of the 3 networks are 3.9, 2.74 and 6.8, respectively. This indicates that there are fewer edges connecting nodes in these 3 networks and that the network is relatively sparse. As a result, many nodes have adjacent edges with only one or two neighboring nodes. This design is very disadvantageous for the K-truss algorithm when finding the core structure of the network. When many nodes in the network are connected with only a few neighboring nodes, it seems more difficult to find the higher-order K-truss structure. In addition, when many nodes in the network are connected with only one neighboring node, using the K-truss algorithm to find the core structure of the network is not effective. Meanwhile, the CSEA algorithm is less effective in these 3 networks because the feature information of the network extracted by the K-truss algorithm in the sparse network is not obvious, which leads to the variational autoencoder model not working well for feature extraction.

However, according to the results, using the NMI metric, the CSEA algorithm is less effective on Cora, Citeseer and DBLP than node2vec, CNM, FUA and Infomap, but higher compared to the other algorithms, so it is in the middle of the range. Using the ACC metric, the CSEA algorithm is only less effective than node2vec on Cora, Citeseer and DBLP (and less effective than the FN algorithm on Cora), which is in the upper-middle range, and shows that the modularity-based algorithm does not have good community division accuracy. Using the F_{same} metric, the CSEA algorithm is less effective than NMF on Cora and Citeseer, and less effective than node2vec on DBLP, which places it in the upper-middle range. Using the Q metric, the CSEA algorithm is less effective than node2vec, FN, CNM, FUA and Infomap on Cora, Citeseer, and DBLP. For YouTube, 1000-level network, the average node degree $\overline{q}$ is 9.39, which is not much denser than Cora, Citeseer and DBLP. The best results are obtained when using ACC and F_{same} metrics, and the results are lower than node2vec and some modularity-based algorithms when using NMI and Q metrics, so it is in the middle of the range. It can be seen that the community division of the CSEA algorithm in the sparser network is better than some algorithms, which is in the middle level and has a certain ability of real community division.

In summary, community detection algorithms using different methods and theories have their own advantages and disadvantages on different real networks. Additionally, according to the experimental results, it is known that the CSEA algorithm is also competitive in community division relative to other algorithms.

Table 8
Comparison of 9 methods in terms of Q on 8 real networks without ground-truth.

Networks	CSEA	FN	SC	NMF	MNDP
Lesmis	**0.548**	0.500	0.478	0.464	0.543
Jazz	**0.443**	0.439	0.315	0.370	0.438
Neural	**0.384**	0.369	0.208	0.156	0.381
Metabolic	0.392	0.400	0.149	0.078	0.380
Email_URV	0.500	**0.502**	0.329	0.302	0.515
Livejournal	0.724	/	0.579	0.331	/
Orkut	0.754	/	/	0.478	/
PGP	0.585	/	/	0.367	/

Networks	Deepwalk	Node2vec	PCA	DSACD	KL
Lesmis	0.219	**0.548**	0.400	0.378	0.326
Jazz	0.429	0.439	0.376	0.315	0.274
Neural	0.324	0.366	0.125	0.214	0.321
Metabolic	0.336	**0.408**	0.118	0.075	0.274
Email_URV	0.478	0.498	0.223	0.142	0.353
Livejournal	0.656	**0.727**	0.346	/	0.487
Orkut	0.502	**0.762**	0.378	/	0.467
PGP	0.655	**0.813**	0.278	/	0.438

4.4.2. Real networks without ground-truth

In real networks without real community labels, because Q does not need to take the real community label as input, only Q can be used as an evaluation metric in this section. The value of Q is less effective than the other algorithms in Section 4.4.1. Based on its characteristics, the CSEA algorithm focuses on the accuracy of community division rather than the number of community divisions. Therefore, Q is meaningful for real networks with unknown community structures. The specific steps are as follows.

First, the parameter p is adjusted to determine the maximum Q value for different networks and recorded. Then, following the approach (Jin, You et al., 2019), we calculate the parameter p for each network by the FUA algorithm and use it as input for the other comparison algorithms. Here we compare only some of the algorithms. According to Section 4.4.1, the modularity-based algorithms divide the real networks into many small sets of communities, which affects the authenticity of the community division, so they are not used as comparison algorithms. The number of community divisions obtained by the FUA algorithm on Lesmis, Jazz, Neural, Metabolic, Email_URV, and PGP are 6, 4, 5, 10, 11 and 18, respectively, and the number of community divisions obtained by the FUA algorithm on the LiveJournal and Orkut networks is too large, so we set it to 8 and 12 according to the size of these 2 networks (Fei et al., 2020).

We use the CSEA algorithm to adjust the parameter p on the basis of the number of community divisions of the FUA algorithm, repeat the experiment 10 times for each adjustment of parameter p and record the best results. The final best parameter p obtained is shown in Fig. 4, and values of is 5, 4, 4, 8, 10, 8, 12, and 18, were obtained. It can be seen that the best parameter p obtained by the CSEA algorithm is smaller than that of the FUA algorithm in small and medium-sized networks, which is also consistent with the situation in 4.4.1. The best parameter p obtained by the CSEA algorithm also matches the set number of community divisions in the denser 1000-level network, and the best parameter p obtained by the CSEA algorithm is not obvious in the sparser 10000-level network, which is also similar to the number of community divisions obtained by the FUA algorithm.

We use the parameter p obtained by the FUA algorithm as input for the other 9 algorithms, while the CSEA algorithm uses the best parameters from Fig. 4. As shown in Table 8, in the FN, SC, and DSACD algorithms, "/" represents the algorithm running for over $18h$ or being unable to handle the memory consumption of this network. In the MNDP algorithm, "/" represents data cases where the relevant network is not given in the literature.

As seen from Table 8, the best results are achieved on Lesmis, Jazz and Neural with values of 0.548, 0.443 and 0.384, respectively, when using the CSEA algorithm. This result also shows that the CSEA

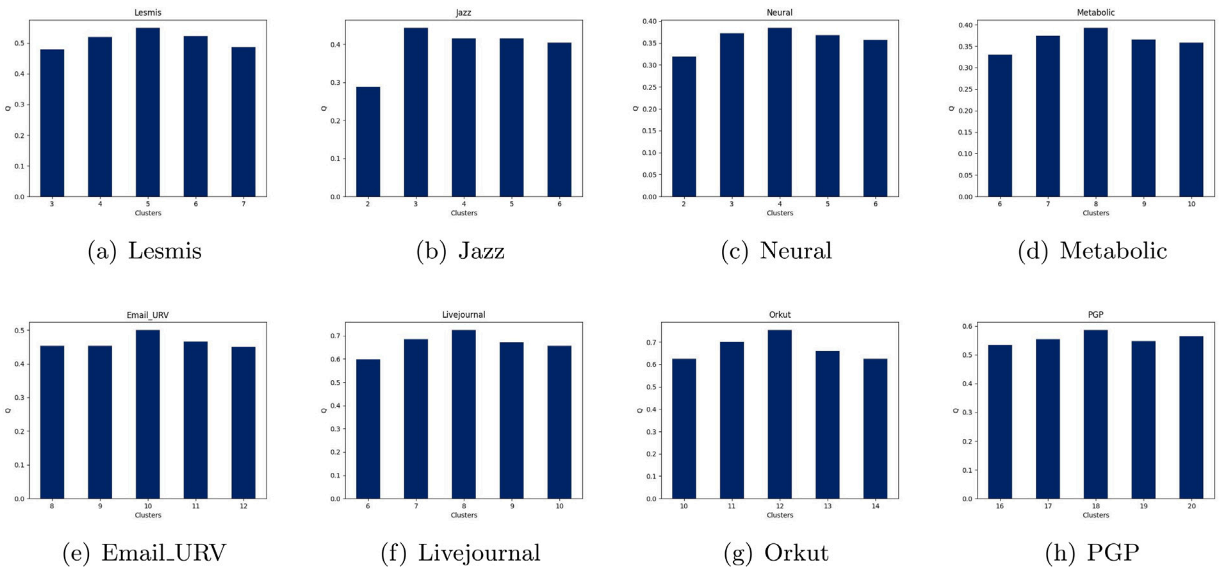

Fig. 4. The CSEA algorithm is used to find the number of community clusters on 8 real networks without ground-truth.

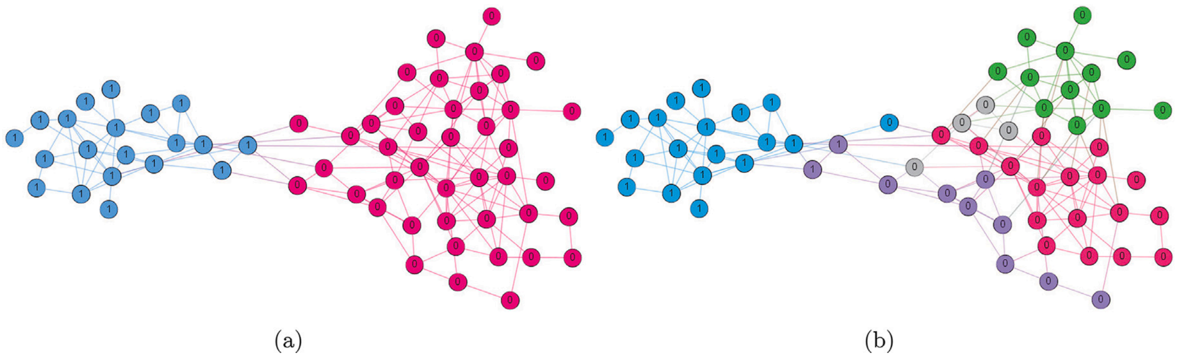

Fig. 5. Dolphins network visualization.

algorithm can be used to obtain better community division in small and medium-sized and dense networks. For Metabolic, the CSEA algorithm is slightly less effective than FN and node2vec. For Email_URV, there is only 0.002 lower value obtained for the CSEA algorithm compared with FN, which indicates that the CSEA algorithm can also obtain a good community division on a medium-sized network with less obvious sparse characteristics. For LiveJournal and Orkut, results achieved using the CSEA algorithm and node2vec are not much different because although they belong to the 1000-level networks, their average node degree $\overline{q}$ is larger, with values of 28.45 and 31.07 respectively, so the CSEA algorithm can find a certain network core structure. For PGP, which is a large sparse network that belongs to the 1000-level networks and its average of node degree $\overline{q}$ is only 4.55. This increases the difficulty of the CSEA algorithm in finding the core structure in the network, so the results are not satisfactory.

In summary, the CSEA algorithm is able to perform more accurate community division in some real networks without ground-truth to some extent.

4.4.3. *Visualization experiment*

From the experimental results in Sections 4.4.1 and 4.4.2, 3 networks Dolphins, Football_12 and Jazz are selected for visualization experiments, as shown in Figs. 5–7. In these figures, (a) represents the results of the CSEA algorithm and (b) represents the results of the FUA algorithm. For the real networks Dolphins and Football_12 with known community structures, the number on each node represents the real community label to which the node belongs. For the real network Jazz, which has an unknown community structure, a larger node in the figure indicates a greater degree of the node. As can be seen from Fig. 5, the community division of the CSEA algorithm on the Dolphins network is exactly the same as the real community, while the FUA algorithm divides 5 communities, and the division accuracy is very low. As can be seen from Fig. 6, the CSEA algorithm on the Football_12 network only divides one node incorrectly (the node marked by red box), and the community division of other parts are exactly the same as the real community, while the FUA algorithm divides 9 communities and the division accuracy is very low. As shown in Fig. 7, the CSEA algorithm divides 3 communities, and the FUA divides 4 communities for the Jazz network.

4.4.4. *Volatility experiment*

Since a deep learning algorithm and K-means algorithm are needed in the CSEA algorithm model, both of the above algorithms have certain randomness, which leads to randomness in the CSEA algorithm. To explore whether the CSEA algorithm has stability, we conduct a volatility experiment. We repeat experiments using the CSEA algorithm 100 times on small and medium-sized networks (Strike, Karate, Dolphins, Polbooks, Football_12, Email_Eu, Polblogs, Cora, Citeseer, Lesmis, Jazz, Neural, Metabolic, Email_URV) and 30 times on large networks (YouTube, DBLP, LiveJournal, Orkut, PGP). The results are shown in Table 9.

As shown in Table 9, the data are more volatile on the small networks. There is a large threshold between the maximum and minimum values, which leads to greater data volatility. For Dolphins, the

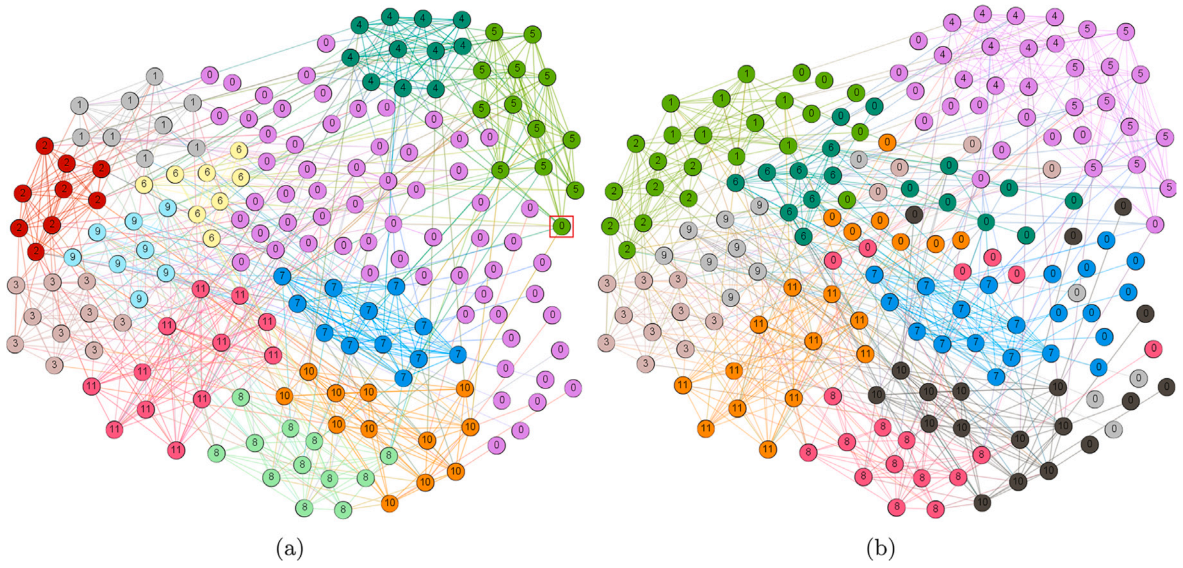

Fig. 6. Football_12 network visualization.

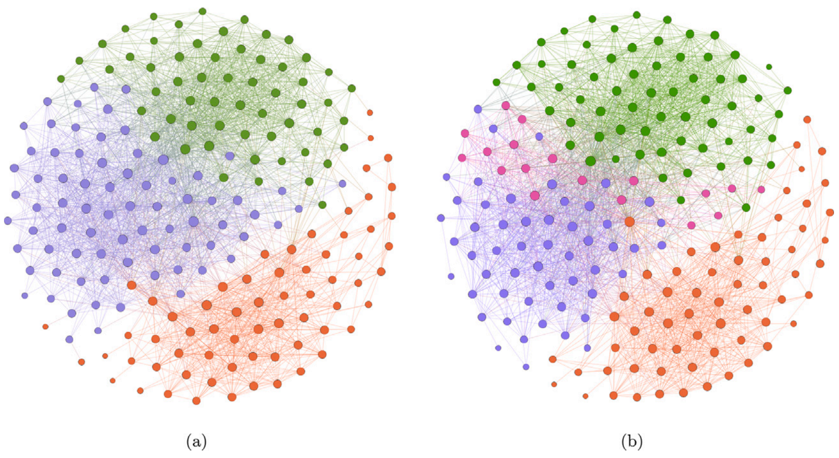

Fig. 7. Jazz network visualization.

Table 9
Volatility analysis of the CSEA algorithm.

Standard deviation ($\times 10^{-3}$)	Q	NMI	F_{same}	ACC
Strike	0.345	66.805	20.767	20.767
Karate	0	0	0	0
Dolphins	20.001	166.810	56.588	61.985
Polbooks	16.241	43.261	32.272	51.181
Football_12	2.885	15.459	19.315	38.325
Email_Eu	15.159	16.893	20.543	27.086
Polblogs	10.629	68.445	33.953	34.586
Cora	16.787	22.395	21.699	25.701
Citeseer	33.273	14.754	19.656	18.237
Youtube	47.658	52.461	50.120	49.772
DBLP	39.849	16.560	22.044	21.130
Lesmis	17.017	/	/	/
Jazz	5.562	/	/	/
Neural	12.352	/	/	/
Metabolic	22.170	/	/	/
Email_URV	13.633	/	/	/
Livejournal	79.572	/	/	/
Orkut	46.108	/	/	/
PGP	31.791	/	/	/

value of NMI can reach a maximum of 1 and a minimum of 0.237. During the 100 experiments, a value of 1 appeared 3 times, values of 0.237 and 0.276 appeared once, and other values were basically stable in the range of 0.8 to 0.9, with some values of approximately 0.6 also appearing. Therefore, the standard deviation is too large mainly because of the large threshold between the maximum and minimum values. There are exceptions, however, perfect divisions are achieved every time on Karate when using the CSEA algorithm. Therefore, the standard deviation for this experiment is 0. In the large and medium-sized dense networks, the data have some volatility, while in the large and medium-sized sparse networks, the data are less volatile than in the dense networks. Additionally, the magnitude of the standard deviation is somewhat related to the number of experiments.

In summary, multiple experiments are required in small and medium-sized networks to reduce the impact of data volatility. The results also show the CSEA algorithm has some stability on large networks as well as can flexibly change the number of repetitions for different networks to achieve the best results.

4.5. Algorithm model efficiency evaluation

We validate the efficiency of the CSEA algorithm using running time as the basic unit, and the basic unit is the second(s). 15 algorithms are compared with the CSEA algorithm on 19 networks. Here, "/" indicates that the algorithm has no running result on that network for the same reason as described in Section 4.4.

Table 10
Comparison of running time with 16 methods on 19 real networks. The basic unit is the second (s).

(a)

Networks	CSEA	FN	NMF	SC	Deepwalk	Node2vec	PCA	DSACD
Strike	0.044	0.018	0.036	0.045	3.665	0.958	0.124	0.777
Karate	0.048	0.077	0.035	0.031	3.725	1.550	0.038	0.776
Dolphins	0.051	0.332	0.039	0.064	3.788	1.877	0.118	0.951
Polbooks	0.059	1.814	0.052	0.086	4.087	2.495	0.052	1.093
Football_12	0.088	12.793	0.086	0.192	4.186	3.599	0.092	1.844
Email_Eu	0.796	858.832	2.267	5.748	18.485	26.790	3.551	132.844
Polblogs	1.452	1447.230	3.969	16.032	21.201	33.020	4.945	479.655
Cora	4.058	6587.824	12.167	60.973	36.709	39.710	23.545	2711.118
Citeseer	5.566	9193.396	21.413	97.335	43.871	45.352	45.333	5656.157
Youtube	29.818	/	84.677	/	127.525	163.869	338.295	/
DBLP	97.445	/	381.636	/	259.405	252.698	1485.699	/
Lesmis	0.058	0.626	0.045	0.067	0.886	1.896	0.117	1.025
Jazz	0.113	19.253	0.080	0.226	1.171	4.784	0.088	1.706
Neural	0.141	103.216	0.139	0.487	1.102	5.807	0.144	4.475
Metabolic	0.223	492.636	0.221	0.983	1.321	8.215	0.260	11.792
Email_URV	0.832	1539.255	1.794	8.207	1.898	18.954	2.548	193.149
Livejournal	23.006	/	62.417	604.382	111.954	156.858	269.329	/
Orkut	48.207	/	136.081	/	176.122	179.689	542.048	/
PGP	64.711	/	243.129	/	187.240	182.686	891.027	/

(b)

Networks	GN	KL	LE	LPA	CNM	FUC	Infomap	Emdot	CDME
Strike	0.014	0.008	0.007	0.009	0.009	0.009	0.008	0.014	0.035
Karate	0.074	0.009	0.008	0.009	0.013	0.010	0.013	0.018	0.056
Dolphins	0.145	0.013	0.007	0.613	0.021	0.015	0.018	0.020	0.088
Polbooks	2.265	0.024	0.009	0.026	0.055	0.036	0.030	0.034	0.280
Football_12	15.461	0.052	0.024	0.038	0.097	0.046	0.039	0.047	0.303
Email_Eu	201.993	1.442	0.196	2.074	4.857	0.930	1.694	1.069	44.293
Polblogs	231.542	2.634	0.053	1.985	7.840	2.126	0.874	/	49.651
Cora	366.566	8.203	0.044	1.568	3.820	2.325	2.477	2.245	19.399
Citeseer	1132.652	12.592	0.186	2.184	3.348	2.676	2.528	/	19.688
Youtube	/	66.458	1.462	33.452	144.937	27.522	18.962	/	606.118
DBLP	/	192.445	2.267	26.892	89.431	36.750	15.065	/	323.826
Lesmis	/	0.015	/	/	/	/	/	/	/
Jazz	/	0.079	/	/	/	/	/	/	/
Neural	/	0.123	/	/	/	/	/	/	/
Metabolic	/	0.268	/	/	/	/	/	/	/
Email_URV	/	1.478	/	/	/	/	/	/	/
Livejournal	/	47.262	/	/	/	/	/	/	/
Orkut	/	94.644	/	/	/	/	/	/	/
PGP	/	131.456	/	/	/	/	/	/	/

Since the CSEA algorithm is a deep learning algorithm, the training part was not covered during the on-line testing. We run the dataset into the already trained model and calculate the time consumption. First, we use C++ to complete the K-truss algorithm to form the similarity matrix, and use Python to complete the deep learning model, then use the K-means algorithm to cluster the resulting low-dimensional feature matrix, and calculate the sum of the two running times to gain the ultimate running time. As shown in Table 10, the running time of the CSEA algorithm is faster than most algorithms in the case of deep learning model training completion, except for some modularity-based algorithms, which have a more obvious efficiency advantage especially in large networks. Meanwhile, these results illustrate that using C++ language to find the K-truss structure in the network can reduce the running time and increase the running efficiency.

5. Conclusion

The core structure in the network is found to have an important impact on community detection, and many algorithms seek communities in the network by analyzing the core structure in the network using only traditional logical analysis techniques. However, these algorithms are generally characterized by high complexity, slow running time, and inconspicuous core structure. To better detect community structures in complex real networks, we propose a novel network core structure extraction algorithm utilized variational autoencoder for community detection (CSEA). The CSEA algorithm combines logic analysis techniques with neural networks, enabling the core structure information in the network to be extracted more significantly. First, we utilize the K-truss algorithm to find the core structure information in complex real networks and combine local information to generate the similarity matrix by similarity mapping. Second, feature dimension reduction and feature extraction of the similarity matrix are carried out by using the variational autoencoder model. Finally, the K-means algorithm is utilized to generate the community information and obtain the final community division. It is proven that the CSEA algorithm has an excellent community division effect in dense complex real networks, especially in small and medium-sized networks, and it can accurately divide the complex real networks with unknown community structure. At the same time, the CSEA algorithm also has certain efficiency advantages over community detection algorithms based on logical analysis techniques. The CSEA algorithm using the K-truss algorithm and variational autoencoders to extract the core structure and form a low-dimensional feature matrix in a complex real network, which is innovative and competitive compared with other community detection algorithms.

For future work, we first intend to modify the similarity matrix generation process so that it can be applied to either a weighted or directed network; this modification is relatively easy since the similarity matrix generation process is a network embedding method. Secondly, we plan to modify the structure of the variational autoencoder to make the algorithm model more suitable for complex real networks with sparse structures. Finally, we propose to optimize the algorithm to make it suitable for even larger complex networks. More recently, we plan to extend CSEA on dynamical networks.

The test for our algorithm are available at: https://github.com/PeterWana/CSEA.

CRediT authorship contribution statement

Rong Fei: Proposed the main idea about the CSEA method, Participated in coding CSEA, Carried out the compared community detection algorithms. **Yuxin Wan:** Participated in coding CSEA, Completed the compare simulation and analyzed the result. **Bo Hu:** Proposed the main idea about the CSEA method, Participated in coding CSEA, Carried out the compared community detection algorithms. **Aimin Li:** Helped to modify the manuscript. **Qian Li:** Helped to modify the manuscript.

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

I have shared the link to my data at https://github.com/PeterWana/CSEA.

Acknowledgments

This work is supported by the National Natural Science Foundation International cooperation and exchange projects of under grant 62120106011, Natural Science Basic Research Program of Shaanxi under grant 2021JM-347, Shaanxi Provincial Department of education Special project under grant No. 21JC026.

References

Amini, A. A., Chen, A., Bickel, P. J., & Levina, E. (2013). Pseudo-likelihood methods for community detection in large sparse networks. *The Annals of Statistics*, *41*(4), 2097–2122. http://dx.doi.org/10.1214/13-AOS1138.

Bahmani, B., Moseley, B., Vattani, A., Kumar, R., & Vassilvitskii, S. (2012). Scalable k-means++. *Proceedings of the VLDB Endowment*, *5*(7), 622–633. http://dx.doi.org/10.48550/arXiv.1203.6402.

Bara'a, A. A., Rada, H. M., Abbas, M. N., & Özdemir, S. (2019). A new evolutionary multi-objective community mining algorithm for signed networks. *Applied Soft Computing*, *85*, Article 105817. http://dx.doi.org/10.1016/j.asoc.2019.105817.

Berlowitz, D., Cohen, S., & Kimelfeld, B. (2015). Efficient enumeration of maximal k-plexes. In *Proceedings of the 2015 ACM SIGMOD International conference on management of data* (pp. 431–444). http://dx.doi.org/10.1145/2723372.2746478.

Bi, X., & Zhao, J. (2021). A novel orthogonal self-attentive variational autoencoder method for interpretable chemical process fault detection and identification. *Process Safety and Environmental Protection*, *156*, 581–597. http://dx.doi.org/10.1016/j.psep.2021.10.036.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(10), P10008. http://dx.doi.org/10.1088/1742-5468/2008/10/P10008.

Boguñá, M., Pastor-Satorras, R., Díaz-Guilera, A., & Arenas, A. (2004). Models of social networks based on social distance attachment. *Physical Review E*, *70*(5), Article 056122. http://dx.doi.org/10.1103/PhysRevE.70.056122.

Bron, C., & Kerbosch, J. (1973). Finding all cliques of an undirected graph. *Communications of the ACM*, *16*(9), 575–577. http://dx.doi.org/10.1145/362342.362367.

Cao, S., Lu, W., & Xu, Q. (2016). Deep neural networks for learning graph representations. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence, vol. 30 no. 1* (pp. 1145–1152).

Cavallari, S., Zheng, V. W., Cai, H., Chang, K. C.-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). http://dx.doi.org/10.1145/3132847.3132925.

Cazals, F., & Karande, C. (2008). A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, *407*(1–3), 564–568. http://dx.doi.org/10.1016/j.tcs.2008.05.010.

Chen, Z., Chen, C., Zhang, Z., Zheng, Z., & Zou, Q.

Choong, J. J., Liu, X., & Murata, T. (2020). Optimizing variational graph autoencoder for community detection with dual optimization. *Entropy*, *22*(2), 197. http://dx.doi.org/10.3390/e22020197.

Clauset, A., Newman, M. E., & Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, *70*(6), Article 066111. http://dx.doi.org/10.1103/PhysRevE.70.066111.

Cohen, J. (2008). Trusses: Cohesive subgraphs for social network analysis. *National Security Agency Technical Report*, *16*(3.1), 1–29.

Cui, G., Zhou, J., Yang, C., & Liu, Z. (2020). Adaptive graph encoder for attributed graph embedding. In *Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery & data mining* (pp. 976–985). http://dx.doi.org/10.1145/3394486.3403140.

Danon, L., Diaz-Guilera, A., Duch, J., & Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, *2005*(09), P09008. http://dx.doi.org/10.1088/1742-5468/2005/09/P09008.

De Santo, A., Galli, A., Moscato, V., & Sperlì, G. (2021). A deep learning approach for semi-supervised community detection in online social networks. *Knowledge-Based Systems*, *229*, Article 107345. http://dx.doi.org/10.1016/j.knosys.2021.107345.

Duch, J., & Arenas, A. (2005). Community detection in complex networks using extremal optimization. *Physical Review E*, *72*(2), Article 027104. http://dx.doi.org/10.1103/PhysRevE.72.027104.

El Kouni, I. B., Karoui, W., & Romdhane, L. B. (2020). Node importance based label propagation algorithm for overlapping community detection in networks. *Expert Systems with Applications*, *162*, Article 113020. http://dx.doi.org/10.1016/j.eswa.2019.113020.

Fei, R., Sha, J., Xu, Q., Hu, B., Wang, K., & Li, S. (2020). A new deep sparse autoencoder for community detection in complex networks. *EURASIP Journal on Wireless Communications and Networking*, *2020*(1), 1–25. http://dx.doi.org/10.1186/s13638-020-01706-4.

Fettal, C., Labiod, L., & Nadif, M. (2022). Efficient graph convolution for joint node representation learning and clustering. In *Proceedings of the Fifteenth ACM International conference on web search and data mining* (pp. 289–297).

Fortunato, S., & Hric, D. (2016). Community detection in networks: A user guide. *Physics Reports*, *659*, 1–44. http://dx.doi.org/10.1016/j.physrep.2016.09.002.

Freeman, L. (2004). The development of social network analysis. *A Study in the Sociology of Science*, *1*(687), 159–167.

Gao, L., Wu, J., Qiao, Z., Zhou, C., Yang, H., & Hu, Y. (2016). Collaborative social group influence for event recommendation. In *Proceedings of the 25th ACM International on conference on information and knowledge management* (pp. 1941–1944). http://dx.doi.org/10.1145/2983323.2983879.

Gillis, N., & Glineur, F. (2012). Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization. *Neural Computation*, *24*(4), 1085–1105. http://dx.doi.org/10.1162/NECO_a_00256.

Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, *99*(12), 7821–7826. http://dx.doi.org/10.1073/pnas.122653799.

Gleiser, P. M., & Danon, L. (2003). Community structure in jazz. *Advances in Complex Systems*, *6*(04), 565–573. http://dx.doi.org/10.48550/arXiv.cond-mat/0307434.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, *27*.

Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 855–864). http://dx.doi.org/10.1145/2939672.2939754.

Guimera, R., Danon, L., Diaz-Guilera, A., Giralt, F., & Arenas, A. (2003). Self-similar community structure in a network of human interactions. *Physical Review E*, *68*(6), Article 065103. http://dx.doi.org/10.1103/PhysRevE.68.065103.

He, D., Song, Y., Jin, D., Feng, Z., Zhang, B., Yu, Z., et al. (2021). Community-centric graph convolutional network for unsupervised community detection. In *Proceedings of the 29th International conference on international joint conferences on artificial intelligence* (pp. 3515–3521). http://dx.doi.org/10.24963/ijcai.2020/486.

Huang, Z., Dou, Z., & Ma, X. (2021). Embedding regularized nonnegative matrix factorization for structural reduction in multi-layer networks. *Applied Soft Computing*, *112*, Article 107781. http://dx.doi.org/10.1016/j.asoc.2021.107781.

Huang, J., Sun, H., Song, Q., Deng, H., & Han, J. (2012). Revealing density-based clustering structure from the core-connected tree of a network. *IEEE Transactions on Knowledge and Data Engineering*, *25*(8), 1876–1889. http://dx.doi.org/10.1109/TKDE.2012.100.

Javed, M. A., Younis, M. S., Latif, S., Qadir, J., & Baig, A. (2018). Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications*, *108*, 87–111. http://dx.doi.org/10.1016/j.jnca.2018.02.011.

Jia, G., Cai, Z., Musolesi, M., Wang, Y., Tennant, D. A., Weber, R. J., et al. (2012). Community detection in social and biological networks using differential evolution. In *Proceedings of the International conference on learning and intelligent optimization* (pp. 71–85). Springer, http://dx.doi.org/10.1007/978-3-642-34413-8_6.

Jin, D., Chen, Z., He, D., & Zhang, W. (2015). Modeling with node degree preservation can accurately find communities. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (pp. 160–167). http://dx.doi.org/10.1177/1461444814555267.

Jin, D., Ge, M., Li, Z., Lu, W., He, D., & Fogelman-Soulie, F. (2017). Using deep learning for community discovery in social networks. In *Proceedings of the 2017 IEEE 29th International conference on tools with artificial intelligence* (pp. 160–167). IEEE.

Jin, D., Li, B., Jiao, P., He, D., & Shan, H. (2019). Community detection via joint graph convolutional network embedding in attribute network. In *Proceedings of artificial neural networks and machine learning - ICANN 2019: Workshop and special sessions* (pp. 594–606). Springer, http://dx.doi.org/10.1007/978-3-030-30493-5_55.

Jin, D., Liu, Z., Li, W., He, D., & Zhang, W. (2019). Graph convolutional networks meet markov random fields: Semi-supervised community detection in attribute networks. In *Proceedings of the AAAI Conference on artificial intelligence, vol. 33 no. 01* (pp. 152–159). http://dx.doi.org/10.1609/aaai.v33i01.3301152.

Jin, D., You, X., Li, W., He, D., Cui, P., Fogelman-Soulié, F., et al. (2019). Incorporating network embedding into markov random field for better community detection. In *Proceedings of the 33rd AAAI Conference on artificial intelligence, vol. 33 no. 01* (pp. 160–167). http://dx.doi.org/10.1609/aaai.v33i01.3301160.

Jin, D., Zhang, B., Song, Y., He, D., Feng, Z., Chen, S., et al. (2020). ModMRF: A modularity-based Markov random field method for community detection. *Neurocomputing*, *405*, 218–228. http://dx.doi.org/10.1016/j.neucom.2020.04.067.

Kernighan, B. W., & Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, *49*(2), 291–307. http://dx.doi.org/10.1002/j.1538-7305.1970.tb01770.x.

Khomami, M. M. D., Rezvanian, A., & Meybodi, M. R. (2016). Distributed learning automata-based algorithm for community detection in complex networks. *International Journal of Modern Physics B. Condensed Matter Physics. Statistical Physics. Applied Physics.*, *30*(8), Article 1650042. http://dx.doi.org/10.1142/S0217979216500429.

Khorasgani, R. R., Chen, J., & Zaiane, O. R. (2010). Top leaders community detection approach in information networks. In *Proceedings of the 4th SNA-KDD Workshop on social network mining and analysis*. Citeseer.

Kim, J., & Lee, J.-G. (2015). Community detection in multi-layer graphs: A survey. *ACM SIGMOD Record*, *44*(3), 37–48. http://dx.doi.org/10.1145/2854006.2854013.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations*. http://dx.doi.org/10.48550/arXiv.1312.6114.

Knuth, D. E. (1993). The stanford graphbase: a platform for combinatorial algorithms. In *Proceedings of the 4th annual ACM-SIAM Symposium on discrete algorithms, vol. 93* (pp. 41–43).

Kong, Y.-X., Shi, G.-Y., Wu, R.-J., & Zhang, Y.-C. (2019). K-core: Theories and applications. *Physics Reports*, *832*, 1–32. http://dx.doi.org/10.1016/j.physrep.2019.10.004.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, *2*(1–2), 83–97. http://dx.doi.org/10.1002/nav.20053.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86. http://dx.doi.org/10.1214/aoms/1177729694.

Latapy, M. (2008). Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theoretical Computer Science*, *407*(1–3), 458–473. http://dx.doi.org/10.1016/j.tcs.2008.07.017.

Li, P.-Z., Huang, L., Wang, C.-D., & Lai, J.-H. (2019). Edmot: An edge enhancement approach for motif-aware community detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 479–487). http://dx.doi.org/10.1145/3292500.3330882.

Li, W., Zhu, H., Li, S., Wang, H., Dai, H., Wang, C., et al. (2021). Evolutionary community discovery in dynamic social networks via resistance distance. *Expert Systems with Applications*, *171*, Article 114536. http://dx.doi.org/10.1016/j.eswa.2020.114536.

Liu, Z., & Ma, Y. (2019). A divide and agglomerate algorithm for community detection in social networks. *Information Sciences*, *482*, 321–333. http://dx.doi.org/10.1016/j.ins.2019.01.028.

Liu, C.-Y., Zhou, C., Wu, J., Hu, Y., & Guo, L. (2018). Social recommendation with an essential preference space. In *Proceedings of the 32ed AAAI Conference on artificial intelligence* (pp. 346–353).

Lu, H., Shen, Z., Sang, X., Zhao, Q., & Lu, J. (2020). Community detection method using improved density peak clustering and nonnegative matrix factorization. *Neurocomputing*, *415*, 247–257. http://dx.doi.org/10.1016/j.neucom.2020.07.080.

Luce, R. D., & Perry, A. D. (1949). A method of matrix analysis of group structure. *Psychometrika*, *14*(2), 95–116. http://dx.doi.org/10.1007/BF02289146.

Luo, L., Fang, Y., Cao, X., Zhang, X., & Zhang, W. (2021). Detecting communities from heterogeneous graphs: A context path-based graph neural network model. In *Proceedings of the 30th ACM International conference on information & knowledge management* (pp. 1170–1180). http://dx.doi.org/10.48550/arXiv.2109.02058.

Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E., & Dawson, S. M. (2003). The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, *54*(4), 396–405. http://dx.doi.org/10.1007/s00265-003-0651-y.

Lyu, T., Zhang, Y., & Zhang, Y. (2017). Enhancing the network embedding quality with structural similarity. In *Proceedings of the 2017 ACM on Conference on information and knowledge management* (pp. 147–156). http://dx.doi.org/10.1145/3132847.3132900.

Mohammadmosaferi, K. K., & Naderi, H. (2020). Evolution of communities in dynamic social networks: An efficient map-based approach. *Expert Systems with Applications*, *147*, Article 113221. http://dx.doi.org/10.1016/j.eswa.2020.113221.

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. http://dx.doi.org/10.1016/j.eswa.2021.115956.

Newman, M. E. (2004). Fast algorithm for detecting community structure in networks. *Physical Review E*, *69*(6), Article 066133. http://dx.doi.org/10.1103/PhysRevE.69.066133.

Newman, M. E. (2006a). Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, *74*(3), Article 036104. http://dx.doi.org/10.1103/PhysRevE.74.036104.

Newman, M. E. (2006b). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, *103*(23), 8577–8582. http://dx.doi.org/10.1073/pnas.060160210.

Newman, M. E. (2013). Community detection and graph partitioning. *Europhysics Letters*, *103*(2), Article 28003. http://dx.doi.org/10.1209/0295-5075/103/28003.

Newman, M. E., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, *69*(2), Article 026113. http://dx.doi.org/10.1103/PhysRevE.69.026113.

Newman, M. E., & Peixoto, T. P. (2015). Generalized communities in networks. *Physical Review Letters*, *115*(8), 088701. http://dx.doi.org/10.1103/PhysRevLett.115.088701.

Ochieng, P. J., Kusuma, W., & Haryanto, T. (2017). Detection of protein complex from protein-protein interaction network using Markov clustering. *Proceedings of the Journal of Physics: Conference Series*, *835*(1), Article 012001. http://dx.doi.org/10.1088/1742-6596/835/1/012001.

Panduranga, N. K., Gao, J., Yuan, X., Stanley, H. E., & Havlin, S. (2017). Generalized model for k-core percolation and interdependent networks. *Physical Review E*, *96*(3), Article 032317. http://dx.doi.org/10.1103/PhysRevE.96.032317.

Park, C., Kim, D., Han, J., & Yu, H. (2020). Unsupervised attributed multiplex network embedding. In *Proceedings of the 34th AAAI Conference on artificial intelligence, vol. 34 no. 04* (pp. 5371–5378). http://dx.doi.org/10.48550/arXiv.1911.06750.

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). http://dx.doi.org/10.1145/2623330.2623732.

Qin, M., Jin, D., Lei, K., Gabrys, B., & Musial-Gabrys, K. (2018). Adaptive community detection incorporating topology and content in social networks. *Knowledge-Based Systems*, *161*, 342–356.

Raghavan, U. N., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, *76*(3), Article 036106. http://dx.doi.org/10.1103/PhysRevE.76.036106.

Reddy, P. K., Kitsuregawa, M., Sreekanth, P., & Rao, S. S. (2002). A graph based approach to extract a neighborhood customer community for collaborative filtering. In *Proceedings of the international workshop on databases in networked information systems* (pp. 188–200). Springer.

Ribeiro, L. F., Saverese, P. H., & Figueiredo, D. R. (2017). Struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 385–394). http://dx.doi.org/10.1145/3097983.3098061.

Roghani, H., Bouyer, A., & Nourani, E. (2021). PLDLS: A novel parallel label diffusion and label selection-based community detection algorithm based on spark in social networks. *Expert Systems with Applications*, *183*, Article 115377. http://dx.doi.org/10.1016/j.eswa.2021.115377.

Rosvall, M., & Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, *105*(4), 1118–1123. http://dx.doi.org/10.1073/pnas.0706851105.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*(3), 93. http://dx.doi.org/10.1609/aimag.v29i3.2157.

Shlens, J. (2014). A tutorial on principal component analysis. *51*, (2), http://dx.doi.org/10.13140/2.1.1593.1684, arXiv preprint arXiv:1404.1100.

Sperlí, G. (2019). A deep learning based community detection approach. In *Proceedings of the 34th ACM/SIGAPP Symposium on applied computing* (pp. 1107–1110). http://dx.doi.org/10.1145/3297280.3297574.

Srinivas, S., & Rajendran, C. (2019). Community detection and influential node identification in complex networks using mathematical programming. *Expert Systems with Applications*, *135*, 296–312. http://dx.doi.org/10.1016/j.eswa.2019.05.059.

Su, X., Xue, S., Liu, F., Wu, J., Yang, J., Zhou, C., et al. (2022). A comprehensive survey on community detection with deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 1–21. http://dx.doi.org/10.1109/TKDE.2021.3118815.

Sun, Z., Sun, Y., Chang, X., Wang, Q., Yan, X., Pan, Z., et al. (2020). Community detection based on the Matthew effect. *Knowledge-Based Systems*, *205*, Article 106256. http://dx.doi.org/10.1016/j.knosys.2020.106256.

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). http://dx.doi.org/10.1145/2736277.2741093.

Tian, F., Gao, B., Cui, Q., Chen, E., & Liu, T.-Y. (2014). Learning deep representations for graph clustering. In *Proceedings of the 28th AAAI Conference on artificial intelligence, vol. 28 no. 1* (pp. 1293–1299).

Ullah, A., Wang, B., Sheng, J., Long, J., Khan, N., & Ejaz, M. (2022). A novel relevance-based information interaction model for community detection in complex networks. *Expert Systems with Applications*, *196*, Article 116607. http://dx.doi.org/10.1016/j.eswa.2022.116607.

Wang, J., Cao, J., Li, W., & Wang, S. (2021). CANE: community-aware network embedding via adversarial training. *Knowledge and Information Systems*, *63*(2), 411–438. http://dx.doi.org/10.1007/s10115-020-01521-9.

Wang, J., & Cheng, J. (2012). Truss decomposition in massive networks. http://dx.doi.org/10.14778/2311906.2311909, arXiv preprint arXiv:1205.6693.

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). http://dx.doi.org/10.1145/2939672.2939753.

Wang, F., Li, T., Wang, X., Zhu, S., & Ding, C. (2011). Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery*, *22*(3), 493–521. http://dx.doi.org/10.1007/s10618-010-0181-y.

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. http://dx.doi.org/10.1016/j.neucom.2021.05.058.

Wang, X., Liu, N., Han, H., & Shi, C. (2021). Self-supervised heterogeneous graph neural network with co-contrastive learning. In *Proceedings of the 27th ACM SIGKDD Conference on knowledge discovery & data mining* (pp. 1726–1736). http://dx.doi.org/10.1145/3447548.3467415.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, *393*(6684), 440–442. http://dx.doi.org/10.1038/30918.

Wu, J., Wang, Y., Shafiee, S., & Zhang, D. (2021). Discovery of associated consumer demands: Construction of a co-demanded product network with community detection. *Expert Systems with Applications*, *178*, Article 115038. http://dx.doi.org/10.1016/j.eswa.2021.115038.

Xie, J., Szymanski, B. K., & Liu, X. (2011). SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *Proceedings of the 2011 IEEE 11th International conference on data mining workshops* (pp. 344–349). IEEE, http://dx.doi.org/10.1109/ICDMW.2011.154.

Xie, Y., Wang, X., Jiang, D., & Xu, R. (2019). High-performance community detection in social networks using a deep transitive autoencoder. *Information Sciences*, *493*, 75–90. http://dx.doi.org/10.1016/j.physa.2016.11.029.

Xin, X., Wang, C., Ying, X., & Wang, B. (2017). Deep community detection in topologically incomplete networks. *Physica A: Statistical Mechanics and its Applications*, *469*, 342–352.

Xu, R., Che, Y., Wang, X., Hu, J., & Xie, Y. (2020). Stacked autoencoder-based community detection method via an ensemble clustering framework. *Information Sciences*, *526*, 151–165. http://dx.doi.org/10.1016/j.ins.2020.03.090.

Yang, L., Cao, X., He, D., Wang, C., Wang, X., & Zhang, W. (2016). Modularity based community detection with deep learning. In *Proceedings of the 29th AAAI Conference on artificial intelligence, vol. 16* (pp. 2252–2258).

Yang, J., & Leskovec, J. (2015). Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, *42*(1), 181–213. http://dx.doi.org/10.1145/2350190.2350193.

Yang, L., Wang, Y., Gu, J., Wang, C., Cao, X., & Guo, Y. (2020). JANE: Jointly adversarial network embedding. In *Proceedings of the 29th International joint conference on artificial intelligence* (pp. 1381–1387).

Yin, H., Benson, A. R., Leskovec, J., & Gleich, D. F. (2017). Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 555–564). http://dx.doi.org/10.1145/3097983.3098069.

You, S., Cho, B. H., Shon, Y.-M., Seo, D.-W., & Kim, I. Y. (2022). Semi-supervised automatic seizure detection using personalized anomaly detecting variational autoencoder with behind-the-ear EEG. *Computer Methods and Programs in Biomedicine*, *213*, Article 106542. http://dx.doi.org/10.1016/j.cmpb.2021.106542.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, *33*(4), 452–473. http://dx.doi.org/10.2307/3629752.

Zhang, Y., Liu, Y., Jin, R., Tao, J., Chen, L., & Wu, X. (2020). GLLPA: A graph layout based label propagation algorithm for community detection. *Knowledge-Based Systems*, *206*, Article 106363.

Zhang, Z., Yang, H., Bu, J., Zhou, S., Yu, P., Zhang, J., et al. (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).

Zheng, Z., Ye, F., Li, R.-H., Ling, G., & Jin, T. (2017). Finding weighted k-truss communities in large networks. *Information Sciences*, *417*, 344–360. http://dx.doi.org/10.1016/j.ins.2017.07.012.

Zhu, J., Chen, B., & Zeng, Y. (2020). Community detection based on modularity and k-plexes. *Information Sciences*, *513*, 127–142. http://dx.doi.org/10.1016/j.ins.2019.10.076.