Hindawi
Security and Communication Networks
Volume 2020, Article ID 8845942, 13 pages
https://doi.org/10.1155/2020/8845942

Research Article

Community Detection Based on DeepWalk Model in Large-Scale Networks

Yunfang Chen,[1] **Li Wang**,[1] **Dehao Qi**,[1] **Tinghuai Ma**,[2] **and Wei Zhang**[1,3]

[1]*School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu 210023, China*
[2]*School of Computer & Software, Nanjing University of Information Science & Technology, Nanjing, Jiangsu 210-044, China*
[3]*Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu 210023, China*

Correspondence should be addressed to Wei Zhang; zhangw@njupt.edu.cn

Received 11 June 2020; Revised 9 September 2020; Accepted 28 October 2020; Published 20 November 2020

Academic Editor: Zhaoqing Pan

The large-scale and complex structure of real networks brings enormous challenges to traditional community detection methods. In order to detect community structure in large-scale networks more accurately and efficiently, we propose a community detection algorithm based on the network embedding representation method. Firstly, in order to solve the scarce problem of network data, this paper uses the DeepWalk model to embed a high-dimensional network into low-dimensional space with topology information. Then, low-dimensional data are processed, with each node treated as a sample and each dimension of the node as a feature. Finally, samples are fed into a Gaussian mixture model (GMM), and in order to automatically learn the number of communities, variational inference is introduced into GMM. Experimental results on the DBLP dataset show that the model method of this paper can more effectively discover the communities in large-scale networks. By further analyzing the excavated community structure, the organizational characteristics within the community are better revealed.

1. Introduction

Recently, complex networks have attracted a great deal of attention in various fields [1–3], including sociology, computer science, mathematics, and biology. Community structure is an important topological property of complex networks [4], where there must be more edges "inside" the community than edges linking vertices of the community with the rest of the graph [5]. The analysis of community is of great significance for us to understand the topological structure of complex networks, mine the hidden laws in networks, predict human's behaviour, and make personalized recommendations [1,2,6,7].

Up to now, a large number of community detection algorithms for complex networks have been proposed. Earlier works mainly include graph partitioning [8,9], modularity-based algorithm [10], spectral methods [11,12], random walk algorithm [13], and so on. For most of the traditional clustering algorithm, it is necessary to provide as input the number of groups and in some cases even their sizes, about which in principle one knows nothing. Instead, one would like an algorithm capable to produce this information in its output. To solve this problem, in 2016, [14] proposed a method of mathematical principle to find the number of communities in the network and the community assignment of nodes. They used Markov chain Monte Carlo (MCMC) [15] to estimate the node assignment and the number of communities. As far as we know, this model is the first random block model to explicitly model the number of communities, and it enables to automatically generate the number of communities [16], extends the definition of spectral graph wavelets to temporal networks by adopting a multilayer framework, and construct a new wavelet filter function. This method is able to select the relevant scales at which communities should be sought automatically. And the study in [17] uses density peak clustering (DPC) to obtain the number of centres as the predefined parameter for nonnegative matrix factorization. In this paper, we adopt the

Gaussian mixture model (GMM) as a clustering model and use the variational inference algorithm based on Dirichlet process to make the number of communities to be preassigned or determined by searching for the best community structure among all candidates.

In addition, most existing approaches toward community detection directly use the observed adjacency matrix and assume that the corresponding edges are highly reliable. This is generally not true to practical applications, since the given graph may be (i) noisy and (ii) loosely related to the clustering problem of interest [18]. Moreover, the sparsity of some networks leads to the complexity of clustering calculation. However, the embedding representation DeepWalk aims to learn the dense and continuous representations of vertices in a low-dimensional space, so that the noise or redundant information can be reduced and the intrinsic structure information can be preserved [19]. As each vertex is represented by a vector containing its information of interest, many iterative or combinatorial problems in network analysis can be tackled by computing mapping functions, distance metrics, or operations on the embedding vectors, and thus avoid high complexity. As the vertices are not coupling any more, it is convenient to apply mainstream parallel computing solutions to large-scale network analysis. As an online graph embedding algorithm, DeepWalk is also scalable, which builds useful incremental results, and is trivially to parallelizable. These qualities make it suitable for a broad class of real-world applications.

In this paper, we propose DeepWalk-based algorithm CD_DVG, which combines DeepWalk embedding representation with a variational Bayesian Gaussian mixture clustering model (VBGMM). Firstly, we use DeepWalk to represent the vertices in the network graph as low-dimensional vectors. Then, GMM is chosen to cluster the nodes in the low-dimensional space. We use variational inference (VI) to replace the expectation maximization (EM) algorithm to infer the posterior for GMM, which named VBGMM. During the clustering process, the model will set the mixture weights of the mixture components which include few nodes to zero, so it can automatically adjust the number of clusters.

Our primary contributions can be summarized as follows:

(1) Combining the method of DeepWalk network representation learning with the variational Bayesian Gaussian mixture model for community detection, experiment shows that the community structure obtained by our method is more in line with the actual results.

(2) Leveraging on low-dimensional embeddings of network topology for community detection, it maintains relationships between vertices. When data are scarce, low-dimensional models generalize better, and speed up convergence and inference.

(3) Experiments show that the proposed model automatically learns the best model configuration for describing similarities in a community by introducing the Bayesian method. Compared with the GMM model, our approach works without specifying the number of communities in advance.

The rest of the paper is arranged as follows. In Section 2, we present the related work of this paper. In Section 3, we present the preliminaries behind the proposed algorithm, including the description of DeepWalk, Gaussian mixture model, and variational inference. In Section 4, we present our approach, CD_DVG. The performance of the proposed algorithm is evaluated in the LFR benchmark and the DBLP dataset, and the results of our experiments are shown in Section 5. Conclusions and suggestions for future work are presented in Section 6.

2. Related Work

The most classic algorithm in community detection is the Girvan–Newman (GN) [20] algorithm, which is a hierarchical clustering algorithm based on edge betweenness partitioning. The algorithm deletes the edge with the largest edge betweenness in the network, and then recalculates the edge betweenness of each edge in the network. This step loops until all the edges in the network are deleted, and finally, a hierarchical clustering tree is established. However, the algorithm requires repeated calculations and high complexity, which is not suitable for large-scale networks.

Graph partitioning partitions the vertices in the graph into different subsets, and guarantees the number of edges between the subsets to be the least, such as the Kernighan–Lin (KL) algorithm [8] and the spectral bisection method based on Laplace matrix [9]. The K-L algorithm is based on greedy features and uses the principles of optimizing the edges within and between communities to partition complex networks into communities. However, this method is difficult to apply to actual application because the algorithm needs to know the size of the community in advance.

The spectral algorithm is to use the eigenvectors of the adjacency matrix or the Laplacian matrix to project the vertices into a new space and cluster them in a new space using traditional clustering methods (such as k-means [4]). Using this method, the network can only be partitioned into two parts at a time, and multiple iterations of the partitioned subcommunities are required, so it has poor speed and clustering effect.

The basic idea of the random walk [21] strategy method is that a "random walker" will always spend a long time in a community because the community structure is relatively close and highly connected. Therefore, starting from a specified vertex, a "random walker" will complete all the vertices within a community in a relatively short time. The method based on modularity [10] improvement uses modularity as the objective function. Meanwhile, it takes the simulate anneal arithmetic (SAA) as the local search method, which improves its accuracy. However, due to the lack of prior knowledge, the algorithm has great limitations.

These traditional topology-based community detection methods usually directly use the observed adjacency matrix, which may contain noise or redundant information. Net-

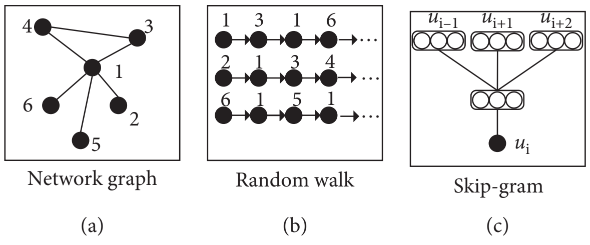


Figure 1: DeepWalk model.

work embedding assigns nodes in a network to low-dimensional representations and effectively preserves the network structure. Recently, a significant amount of progress has been made toward this emerging network analysis paradigm, such as [22–25]. There exists a kind of network embedding method which preserves structure information. Considering a network with only topology information, many network analysis tasks, such as community detection, can be conducted in the original network space. However, for large networks, the traditional network representation poses several challenges to network processing and analysis, such as high computational complexity. Motivated by this, attempts are proposed to preserve rich structural information on network embedding, from nodes and links [26] to neighborhood structure [27], high-order proximities of nodes [28], and community structures [29].

The embedding method is common in clustering in recent years, but few literature studies combine it with variational inference. Generally, k-means is applied to the learned latent vector with giving a priori. A Gaussian mixture model empowered with Bayesian can provide insights in terms of uncertainty and exploit nonlinearities which result in better performance in comparison to state-of-the-art community detection methods [30]. Additionally, Bayesian inference solves the problem where the number of communities is uncertain [2], uses variational inference for community detection, and obtains a good result, which first convolves on the network, learning structural and nodal features in the process. However, our paper uses the DeepWalk method to get the features of the nodes. Relationships between nodes in higher dimensions are preserved in lower dimensions.

3. Preliminaries

3.1. DeepWalk. DeepWalk [27] learns a social representation of a network by truncating random walks and outputs the vector representation of the vertices in the network, and its input is a network graph $G(V, E)$. The process of DeepWalk is shown in Figure 1: firstly, generating sequences of network graph vertices by random walk; secondly, inputting the sequence of vertices into skip-gram language model. Through continuous training of the skip-gram model, the vector representation matrix $\Phi \in \mathbb{R}^{|V|\times d}$ (d is the dimension of embedding space) of each vertex in the network is finally obtained.

3.1.1. Random Walk. This model is exploited to generate random walks over a network. We assume that a random walk rooted at vertex v_i as W_{v_i}. It is a stochastic process with random variables $W_{v_i}^1, W_{v_i}^2, \ldots, W_{v_i}^k$, such that $W_{v_i}^{k+1}$ is a vertex chosen at random from the neighbors of vertex v_k. By regarding a node as a word, we can regard a random path as a sentence, and the node neighborhood can be identified by cooccurrence rate as in Word2Vector [19].

3.1.2. Skip-Gram. It is a language model that maximizes the cooccurrence probability among the words that appear within a window in a sentence [31]. Instead of predicting the current word based on the context, skip-gram tries to maximize the classification of a word based on another word in the same sentence. More precisely, this algorithm uses each current word as an input to a log-linear classifier with continuous projection layer, and predicts words within a certain range (such as w) before and after the current word. So, the problem is to estimate the likelihood:

$$\log \Pr\left(\{v_{i-w}, \cdots, v_{i-1}, v_{i+1}, \ldots, v_{i+w}\} | v_i\right). \tag{1}$$

3.2. Gaussian Mixture Model. The variational Bayesian Gaussian mixture model (VBGMM) is selected as a clustering algorithm in this paper. Since VBGMM is an extension of the basic Gaussian mixture model, this section describes the Gaussian mixture model (GMM).

GMM uses the combination of finite Gaussian probability density functions to estimate the distribution of samples. The purpose of GMM clustering is to maximize the probability density of the samples. Each Gaussian function is a cluster, and each sample will be clustered into the Gaussian function which has the largest probability density [32]. The GMM is defined as

$$p(x) = \sum_{i=1}^{K} \alpha_i \cdot p(x | \mu_i, \Sigma_i), \tag{2}$$

where x is a d-dimension data, and K is the number of "mixture components," each of which corresponds to a Gaussian distribution. α_i denotes an estimate of the weight of the i^{th} Gaussian in the mixture, and $\sum_{i=1}^{K} \alpha_i = 1$. μ_i denotes the mean value (average vector) of the i^{th} Gaussian, Σ_i is the covariance matrix of the i^{th} Gaussian, and $p(x | \mu_i, \Sigma_i)$ is i^{th} Gaussian probability density function.

The GMM is a classical clustering algorithm. It uses a Gaussian distribution as a parametric model, assuming that the data obey a mixture Gaussian distribution and estimating the parameters using the EM algorithm [33].

3.3. Variational Inference. The goal of variational inference is to approximate a conditional density of latent variables given observed variables. The key idea is to solve this problem with optimization. A family of densities is used over the latent variables, parameterized by free variational parameters. The optimization finds the member of this family, i.e., the setting of the parameters, which is the closest in KL divergence to the conditional of interest. The fitted variational density then serves as a proxy for the exact conditional density [34].

Let $x = \{x_1, x_2, \ldots, x_n\}$ be a set of observed variables and $z = \{z_1, z_2, \ldots, z_m\}$ be a set of latent variables, with joint density $p(z, x)$. The constants are omitted, such as hyperparameters, from the notation.

The inference problem is to compute the conditional density of the latent variables given the observations, $p(z|x)$. This conditional density can be used to produce point or interval estimates of the latent variables, form predictive densities of new data, and more.

In variational inference, $q(z)$ is a candidate approximation to the exact conditional probability density $p(z|x)$. The goal is to find the best candidate, the one closest in KL divergence to the exact conditional, which is expressed as follows:

$$q^*(z) = \arg\min_{q(z)} \mathrm{KL}(q(z) \| p(z|x)), \tag{3}$$

where $\mathrm{KL}(q(z)\|p(z|x)) = E[\log q(z)] - E[\log p(z,x)] + \log p(x)$. Once found, $q^*(\cdot)$ is the best approximation of the conditional. Because we cannot compute the KL, we optimize an alternative objective that is equivalent to the KL up to an added constant:

$$\mathrm{ELBO}(q) = E[\log p(z,x)] - E[\log q(z)]. \tag{4}$$

Maximizing the ELBO is equivalent to minimizing the KL divergence.

4. CD_DVG Algorithm

4.1. Algorithm Description. The algorithm in this paper is based on the network structure for community detection, named CD_DVG (community detection by the DeepWalk and variational Bayesian Gaussian mixture model) algorithm. Its input is an undirected and unweighted graph $G(V, E, A)$, where $V = \{v_1, v_2, \ldots, v_n\}$ represents the vertex set, $E = \{e_{ij} | i, j \in V, i \neq j\}$ refers to the edge set, and $A \in \mathbb{R}^{|V|\times|V|}$ is the adjacency matrix. Considering the complexity and sparsity of large-scale networks, we use the DeepWalk algorithm for dimensionality reduction. The output of this step is a vector representation matrix $\Phi \in \mathbb{R}^{|V|\times d}$, where a row in the matrix represents the feature of a vertex. Then, the representation matrix Φ is input into the VBGMM to obtain the clustering result of the network graph. Here, we consider each cluster as a community and each vertex in the cluster as a community member. The overview of the proposed algorithm in this paper is given in Figure 2.

4.2. Vector Representation. We choose DeepWalk as the vector representation algorithm, which consists of a random walk generator and a skip-gram update procedure. First, the generator randomly samples a random vertex v_i as the root of the random walk W_{v_i}. A walk sample uniformly from the neighbors of the last vertex visited until the maximum length t is reached. The generator specifies the number of random walks γ of length t to start at each vertex. In practice, the length of the random walk sequence depends on the number of vertices (or the size of the graph). We should set the length of the walk to be larger if the number of vertices in the graph is relatively large. Then, the skip-gram model is chosen for training to get and update the vector representation of the vertices. The goal is to learn a latent representation, not only a probability distribution of node cooccurrences, and so we introduce a mapping function Φ: $v \in V \mapsto \mathbb{R}^{|V|\times d}$, where d is the dimension of embedded space. This mapping Φ represents the latent social representation associated with each vertex v in the graph. So, equation (1) can be optimized as follows:

$$\Pr(\{v_{i-w}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+w}\} | \Phi(v_i)). \tag{5}$$

In Algorithm 1, the input of the algorithm is window size w (predicted words range), embedding dimension d, walks per vertex γ, walk length t, and graph G. Line 3 specifies that each vertex needs to generate γ random walks. In each loop of the process of generating a random walk sequence, the algorithm first shuffles all the vertices in the network graph G and then traverses each vertex in the set V and generate a random walk sequence $|W_{v_i}| = t$ for each vertex v_i . Finally, the walk sequence W_{v_i}, the vector representation matrix Φ, and the window size w are input into the skip-gram algorithm (line 7).

The skip-gram model, described in Algorithm 2, is divided into two steps. The first is to build the model, and the second is to obtain the vector representation of the vertex through the model. The process of the skip-gram algorithm is very similar to that of the self-encoder, that is, constructing a neural network based on the training data. When the model is trained, we do not need to use this model to deal with new tasks, but rather the parameters, we need to get in the training model, such as the weight matrix of the hidden layer. In this paper, the weight matrix is the vector representation matrix of the vertices.

In line 3 of Algorithm 2, the loss function is defined as the conditional probability of the output vertex, usually in logarithmic form, $J(\Phi) = -\log \Pr(u_k | \Phi(v_j))$. The next step is to derive the loss function. By this way, we get the update rule of the vector representation matrix: $\Phi = \Phi - \alpha(\partial J / \partial \Phi)$. From this update rule, we can see that the computational complexity is very high for a large corpus. In order to speed up the training process, the algorithm uses the hierarchical softmax method.

Hierarchical softmax decomposes the conditional probability by assigning each vertex to the leaf vertex of the binary tree and encoding the vertex using the Huffman code. If the vertex appears relatively frequently, its path will be set to be shorter. Suppose the path from the root vertex b_0 to

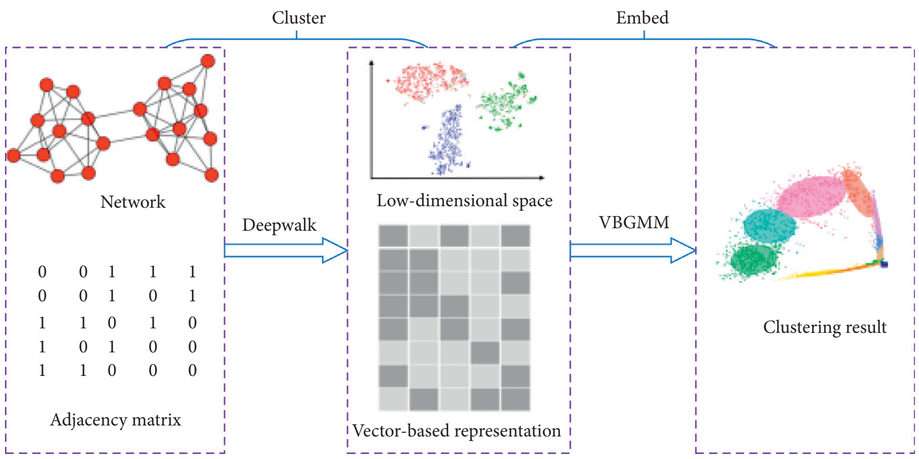


FIGURE 2: The flowchart of CD_DVG algorithm.

```
Input: $\mathbf{G}(\mathbf{V},\mathbf{E})$, $w$, $d$, $\gamma$, $t$
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V|\times d}$
(1) Initialize $\Phi$
(2) Build a Binary Tree $T$ from $V$
(3) for $i = 0$ to $\gamma$ do
(4)     $\mathrm{O} = \mathbf{shuffle}(\mathbf{V})$
(5)     for each $\mathbf{v_i} \in \mathrm{O}$ do
(6)         $\mathbf{W}_{\mathbf{v_i}} = \mathrm{RandomWalk}(\mathbf{G}, \mathbf{v_i}, \mathbf{t})$
(7)         skip-gram $(\Phi, \mathbf{W}_{\mathbf{v_i}}, \mathbf{w})$
(8)     end
(9) end
```

ALGORITHM 1: Vector representation process of vertex.

```
Input: $\Phi, \mathbf{W}_{\mathbf{v_i}}, \mathbf{w}$
Output: the updated $\Phi$
(1) for each $\mathbf{v_j} \in \mathbf{W}_{\mathbf{v_i}}$ do
(2)     for each $\mathbf{u_k} \in \mathbf{W}_{\mathbf{v_i}}[\mathbf{j}-\mathbf{w}:\mathbf{j}+\mathbf{w}]$ do
(3)         $\mathbf{J}(\Phi) = -\mathbf{logPr}(\mathbf{u_k}|\Phi(v_j))$
(4)         $\Phi = \Phi - \alpha\partial\mathbf{J}/\partial\Phi$
(5)     end
(6) end
```

ALGORITHM 2: The updating process of Φ.

another vertex u_k is a sequence of tree vertices $(b_0, b_1, \ldots, b_{\log|V|})$, where $b_{\log|V|}$ indicates the vertex u_k, so there is

$$\Pr\left(u_k|\Phi\left(v_j\right)\right) = \prod_{l=1}^{\log|V|} \Pr\left(b_l|\Phi\left(v_j\right)\right), \tag{6}$$

where $\Pr(b_l|\Phi(v_j)) = (1 + e^{-\Phi(v_j)\cdot\Psi(b_l)})/1$, $\Psi(b_l)$ is the latent representation of the parent vertex of b_l.

4.3. Vertex Clustering. The ultimate goal of this section is to cluster the vertices in the network. In the previous section, we have obtained the vector representation matrix Φ for the network. In the matrix Φ, we consider each vertex as a sample and the vector representation of the vertex as a sample feature. So, Φ can be treated as a sample set $D = \{x_1, x_2, \ldots, x_n\}$, where each element x_i in set D is a column vector of dimension d, and $D = \Phi \in \mathbb{R}^{|V|\times d}$. Sample set D is treated as the input of the clustering model VBGMM, which is the extension of GMM (Section 3.2). The difference is that the VBGMM adopts the variational inference algorithm [18] to train the model parameters, while GMM adopts the EM algorithm.

Assume the sample set $D = \{x_1, x_2, \ldots, x_n\}$ is generated by the above process, and the random variable $z_j \in \{1, 2, \ldots, K\}$ represents the Gaussian mixture component of samples generated by the set D. Here, the values of z_j

are unknown. The prior probability $P(z_j = i)$ of z_j corresponds to $\alpha_i (i = 1, 2, \ldots, K)$. According to the Bayesian theorem, the posterior probability distribution of z_j is shown as follows:

$$\begin{aligned}\gamma_{ji} = p\left(z_j = i|tx_j\right) &= \frac{p\left(z_j = i\right) \cdot p\left(x_j|z_j = i\right)}{p\left(x_j\right)} \\ &= \frac{\alpha_i p\left(x_j|\mu_i, \Sigma_i\right)}{\sum_{l=1}^{K} \alpha_l p\left(x_j|\mu_l, \Sigma_l\right)},\end{aligned} \tag{7}$$

where $p(z_j = i|x_j)$ is the posterior probability of the i^{th} Gaussian mixture component to generate x_j. In other words, in clustering, this equation indicates the probability that the sample belongs to the category z_j. For the convenience of description, we abbreviate $p(z_j = i|x_j)$ into $\gamma_{ji} (i = 1, 2, 3, \ldots, K)$, where γ_{ji} denotes the probability that the j^{th} observation comes from the i^{th} cluster. Usually, we choose the most probable mixture component as the final cluster result of sample data points, and the sample data x_j belong to cluster λ_j, as follows:

$$\lambda_j = \underset{i\in\{1,2,\ldots,K\}}{\arg\max}\, \gamma_{ji}. \tag{8}$$

We choose the Gaussian mixture distribution as a prior probability model. The mean parameters are drawn independently from a common prior $p(\mu_i)$, which we assume to be a Gaussian $N(0, \sigma^2)$; the prior variance σ^2 is a hyperparameter. In order to determine the Gaussian distribution $p(z_j = i|x_j)$, we turn to the variational density $q(\mu_i; m_i, s_i^2)$ of the i^{th} mixture component. In this paper, we use a variational inference algorithm to update the parameters. The variational update for the j^{th} cluster assignment (i.e., probability that the j^{th} observation comes from the i^{th} cluster) is described as follows:

$$\gamma_{ji} \propto \exp\left\{E\left[\mu_i; m_i, s_i^2\right]x_j - \frac{E\left[\mu_i^2; m_i, s_i^2\right]}{2}\right\}, \tag{9}$$

where $E(\cdot)$ is expectation function. Notice it is only a function of the variational parameters for the mixture components.

The updates for the variational mean and variance are defined as follows:

$$\begin{aligned} m_i &= \frac{\sum_j \gamma_{ji} x_j}{\left(1/\sigma^2\right) + \sum_j \gamma_{ji}}, \\ s_i^2 &= \frac{1}{\left(1/\sigma^2\right) + \sum_j \gamma_{ji}}. \end{aligned} \tag{10}$$

Gaussian mixture model based on variational inference algorithm includes two weight distribution prior methods in the acquisition of parameters: One is the finite mixture model of Dirichlet distribution, the other is the infinite mixture model of Dirichlet process [19]. In this experiment, we choose the latter method to estimate the parameters of the model. In practice, the Dirichlet process inference algorithm is approximated and uses a truncated distribution with a fixed maximum number of components (called the stick-breaking representation). The number of components actually used almost always depends on the data. When the concentration parameter (weight_concentration_prior) is set sufficiently small and the number of mixture components is set larger than the actual number of mixture components required by the model, some mixture weights will approach to 0, so that the model can automatically select the appropriate mixture component. The overall flow of the variational Bayesian Gaussian mixture clustering algorithm is described as follows (Algorithm 3).

The lines 2 to 10 are based on the variational inference of the Dirichlet process algorithm to infer the model parameters, and iteratively update. When the parameter estimation algorithm stop condition is satisfied, the cluster of the sample data points is determined according to the Gaussian mixture distribution. It should be noted that the parameter $k_{\max}$ of the model only indicates the maximum number of mixture components that may be used. The number of mixture components required by the model is often less than $k_{\max}$ in actual operation and is depending on the specific quantity of sample data.

5. Experiments

In this part, the performance of our algorithm is verified on the real-world network and LFR benchmark network, respectively. First, we analyze the proposed method's parameters, and then use the analysis to find the most suitable parameters for each network, so as to get the best result of the algorithm. Compared methods are as follows: Louvain and GN. The hardware environment for the experiment is as follows: Intel (R) Core (TM) i5-9300H CPU, 2.40 GHz, and 8 GB RAM. Python 3.6 was used for the software environment.

5.1. Measures and Networks

5.1.1. Measures. In this paper, we apply normalized mutual information (NMI) to evaluate the performance of different algorithms. NMI is a widely used similarity measure metric, which is originated from information theory and proved to be reliable. NMI characterizes the similarity between the true community partition and the partition obtained by the algorithm. Normalized mutual information is defined as

$$\text{NMI}(Y, C) = \frac{2 \times \text{MI}(Y; C)}{[H(Y) + H(C)]}, \tag{11}$$

where Y are class labels (that is truth-label), C are cluster labels, $H(\cdot)$ is entropy, and MI (Y, C) is mutual information of Y and C. The NMI ranges from 0 to 1, and the higher it is, the more accurate the algorithm is.

We use $F1$-measure to analyze the optimal parameters of the network. As a comprehensive evaluation metric, F-measure is the harmonic mean of the precision and the recall, which is calculated as follows:

```
Input: sample data set $\mathbf{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$; the prior variance of component variance $\sigma^2$; the maximum number of mixture components $\mathbf{k}_{\mathbf{max}}$; concentration parameter weight_concentration_prior.
Output: cluster partition $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_K\}$ ($K$-categorical)
(1)  Initializing variational parameters $\mathbf{m} = \mathbf{m}_{1:K}$, $\mathbf{s}^2 = \mathbf{s}^2_{1:K}$, $\gamma = \gamma_{1:n}$
(2)  while ELBO has not converged do
(3)      for $j = 1$: $n$ do
(4)          set $\gamma_{\mathbf{ji}} \propto \exp\{\mathrm{E}[\mu_{\mathbf{i}}; \mathbf{m}_{\mathbf{i}}, \mathbf{s}_{\mathbf{i}}^2]\mathbf{x}_{\mathbf{j}} - \mathrm{E}[\mu_{\mathbf{i}}^2; \mathbf{m}_{\mathbf{i}}, \mathbf{s}_{\mathbf{i}}^2]/2\}$
(5)      end
(6)      for $i = 1$: $\mathbf{k}_{\mathbf{max}}$ do
(7)          set $\mathbf{m}_i \leftarrow \sum_j \gamma_{ji} x_j / (1/\sigma^2) + \sum_j \gamma_{ji}, \mathbf{s}_i^2 = 1/(1/\sigma^2) + \sum_j \gamma_{ji}$
(8)      end
(9)      Compute ELBO $(\mathbf{m}, \mathbf{s}^2, \gamma)$
(10) end
(11) $\mathbf{C}_i = \varnothing\{1 \le \mathbf{i} \le \mathbf{K}\}$
(12) for $j = 1$: $n$ do
(13)     Calculate $\lambda_{\mathbf{j}}$ for sample data point $\mathbf{x}_{\mathbf{j}}$
(14)     $\mathbf{C}_{\lambda_j} = \mathbf{C}_{\lambda_j} \cup \{x_j\}$
(15) end
```

Algorithm 3: Gaussian mixture clustering with variational inference.

$$F = \frac{(\alpha^2 + 1)P \times R}{\alpha^2 (P + R)}, \quad (12)$$

where P represents the precision, which is the ratio of the number of related articles retrieved to the total articles retrieved. R is recall, which is the ratio of the number of related articles retrieved to the actual related articles in the literature library. When $\alpha = 1$, it becomes the most common $F1$-measure, $F1 = 2P \times R/P + R$, which is used in our paper. Generally, the higher the $F1$-measure, the better the algorithm performance.

5.1.2. LFR Benchmark Networks. The LFR dataset is used in our experiment, which is a type of computer-generated network with predefined tunable parameters. The networks possess real-world characteristics and are widely applied in various overlapping network community detection algorithms. The network parameters are shown in Table 1. Table 2 shows the parameter settings for the LFR benchmark networks in this experiment.

5.1.3. DBLP Network. Taking the DBLP dataset as an example, we give the process of parameter analysis. In DBLP, the elements of an article include author, title, pages, year, volume, journal, and number. We extract collaboration data from five areas in the DBLP dataset: database (SIGMOD, ICDE, VLDB, EDBT, PODS, ICDT, DASFAA, SSDBM, and CIKM), data mining (KDD, ICDM, SDM, PKDD, and PAKDD), artificial intelligence (IJCAI, AAAI, NIPS, ICML, ECML, ACML, IJCNN, UAI, ECAI, COLT, ACL, and KR), and computer vision (CVPR, ICCV, ECCV, ACCV, MM, ICPR, ICIP, and ICME), which amount to 30,422 articles. Considering each article as a vertex, 30,422 articles constitute a network graph of 30,422 vertices.

Table 1: LFR parameter and description.

Parameter	Description
N	Number of nodes
k	Average degree of nodes
max$k$	Maximum degree
min$C$	Minimum for the community sizes
max$C$	Maximum for the community sizes
t_1	Minus exponent for the degree sequence
t_2	Minus exponent for the community size distribution
mu	Mixing parameter
om	Number of memberships of the overlapping nodes
on	Number of overlapping nodes
C	Average clustering coefficient

Table 2: Parameter settings of our experiment.

Network	N	k	t_1	t_1	min$C$	max$C$	mu
LFR1	1000	20	2	1	10	50	0.2
LFR2	1000	20	2	1	20	100	0.2
LFR3	5000	20	2	1	10	50	0.2
LFR4	5000	20	2	1	20	100	0.2

5.2. Parameter Analysis. The experiment in this section consists of two stages. First, this algorithm represents the vertices in the network as vectors. This stage mainly uses the DeepWalk network embedding algorithm. And there are four parameters involved, including the walk sequences length t, walks per vertex γ, the dimension d of the vector representation space, and the window size w.

In the second stage, this algorithm clusters the vector representation of the vertices. In this stage, the VBGMM is used. The parameters mainly include the cluster number k and the concentration parameter weight_concentration_prior.

5.2.1. The Parameters during Network Embedding. In order to evaluate the impact of the above four parameters on the performance of the model, we adjust the parameters one by one. We fixed three parameters at a time and set different values for another parameter. In this way, we recorded the results of the vertex vector representation. Then, clustering is performed, and the clustering effect is measured by the $F1$-measure.

In Figure 3, we fix the walking length t, the dimension d of vertices vector representation, and the window size w, and change the number of walks to get $F1$-measure. Then, by changing the window size, $F1$ curve can be obtained. From this figure, we can see that the learning algorithm achieves the best clustering effect and gradually becomes stable when the number of walks γ reaches 60. Moreover, when the window size w is equivalent to 15, the $F1$-measure of clustering results is larger than that in other window sizes. In order to achieve a more accurate community partitioning result, we set the window w to 15, and the number of walks γ to 60. In the experiment, we did not take a larger number of walks due to the model runtime.

In Figure 4, we set different dimensions of vector representations and different length of the walks, and calculate the corresponding $F1$-measure. Obviously, the $F1$ curve is the highest when the walk length t is 60, and the curve is the lowest when the walk length is 20, which means that when the walk length is small, the vertex's walk sequence may not have traversed all the vertices associated with the starting vertex. Also, when the vertex vector representation dimension d is less than 128, the $F1$-measure is positively correlated with d. However, when the dimension is greater than 128, the four $F1$ curves have a downward trend. Therefore, in the following experiment, we set the walk sequence length to 60 and the dimension of vector representation to 128.

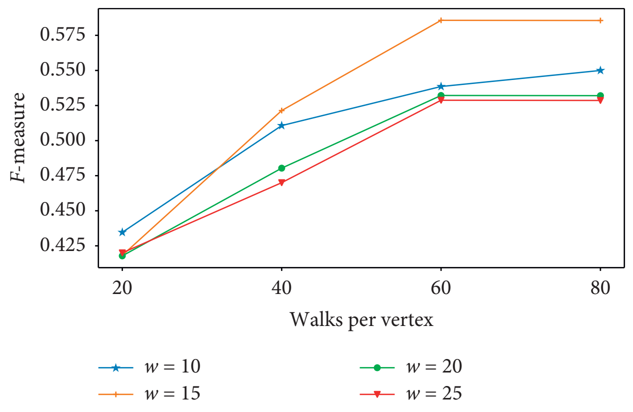


FIGURE 3: The $F1$-measure by changing walks and window size.

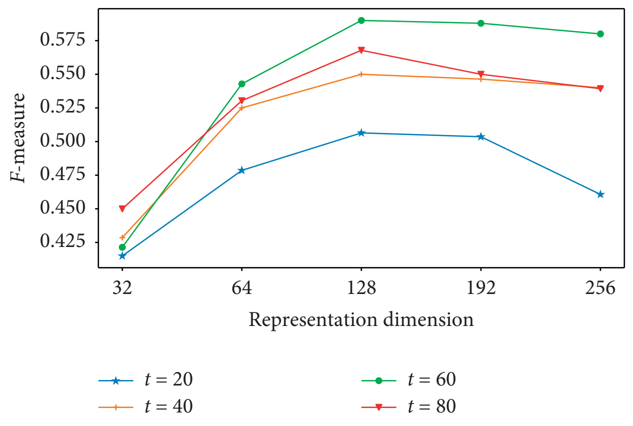


FIGURE 4: The $F1$-measure by changing dimension and walk length.

5.2.2. The Parameters during Clustering. Considering the variational Bayesian Gaussian mixture model for clustering, the concentration parameter is mainly adjusted in this experiment. When the value of the concentration parameter is small enough, by setting the number of clusters $k_{\max} > k_{real}$ (k_{real} is the real number of communities), the model will automatically adjust the clustering result to fit the real number of clusters. In the follow experiment, we set the concentration parameter weight_concentration_prior = 0.001.

5.3. Algorithm Comparison. In the previous chapter, we analysed the optimal parameters of our algorithm. We use the parameter analysis method (Section 5.2) to analyze network parameters and select the optimal case. Because we choose NMI as the indicator of community detection to compare with other methods, here we choose NMI indicator to plot the linear graph when analyzing the optimal parameters.

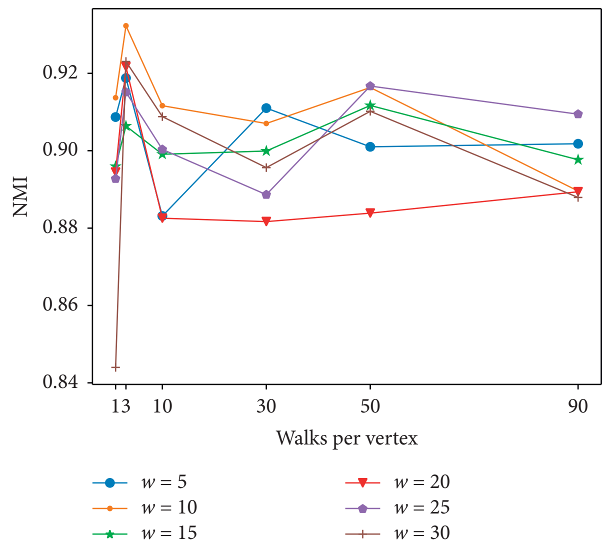
Figures 5(a) and 5(b) show linear graphs of parameter changes in the LFR1 network. It can be seen that the curve of $w = 10$ is almost greater than the value of all curves. When the parameter $r = 3$, the value of all curves is the maximum. When $r > 3$, although the value fluctuates slightly, it presents a downward trend. In Figure 5(b), the trend is very obvious. When $d = 8$, the value of NMI reaches the maximum. With the increase of dimension, NMI gradually decreases. When $t = 80$, NMI reaches the maximum. According to the above four-fold line graph, we finally get the optimal parameter as follows:

LFR1: $r = 3$, $w = 10$, $d = 8$, and $t = 80$; LFR2: $r = 10$, $w = 5$, $d = 8$, and $t = 40$; LFR3: $r = 50$, $w = 30$, $d = 16$, and $t = 80$; LFR4: $r = 30$, $w = 30$, $d = 16$, and $t = 60$.

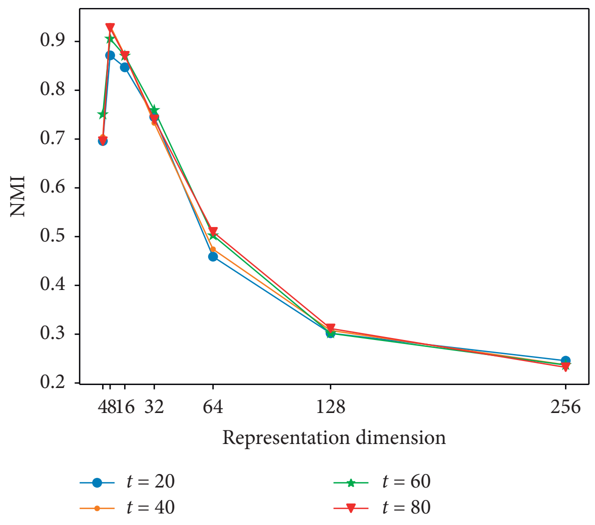
The NMI of our algorithm and other algorithms is compared as follows:

The following experiments use the parameters obtained from the previous analysis.

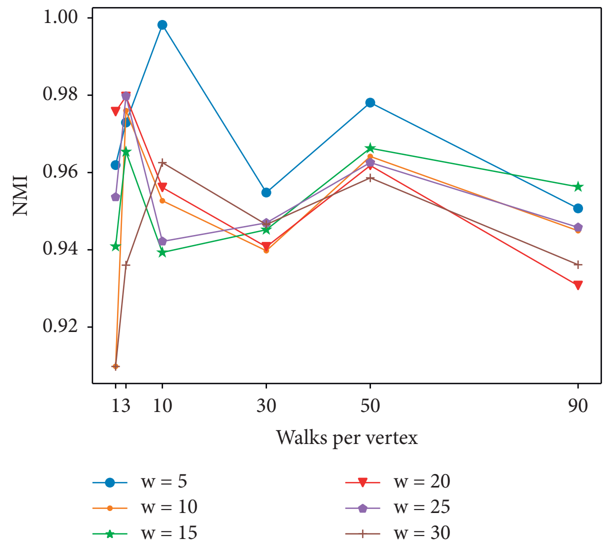
In Table 3, we can see that in the first two networks, the result of our algorithm is equal to or slightly lower than the GN algorithm, but always superior to the Louvain algorithm. In the latter two networks, our algorithm is always higher than the Louvain algorithm and is very close to 1. GN algorithm gets the best result on LFR1 and LFR2, but it took about 5 hours on our device. Moreover, with the increase of the network scale, the time required also exploders. After



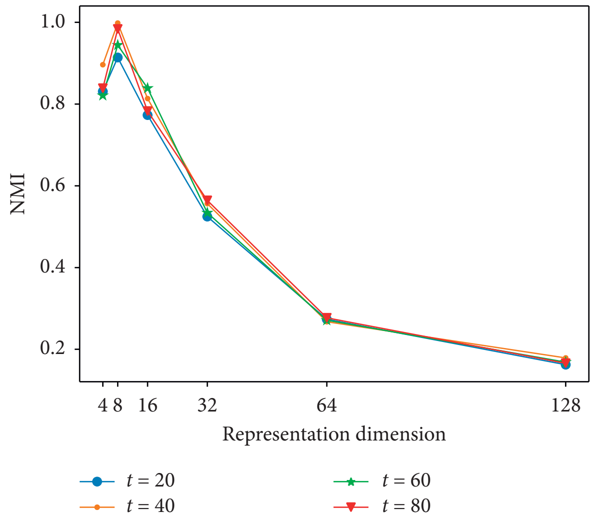
(a)



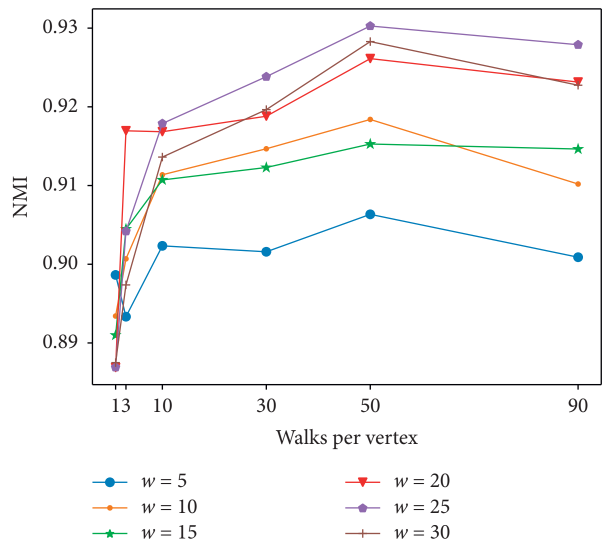
(b)



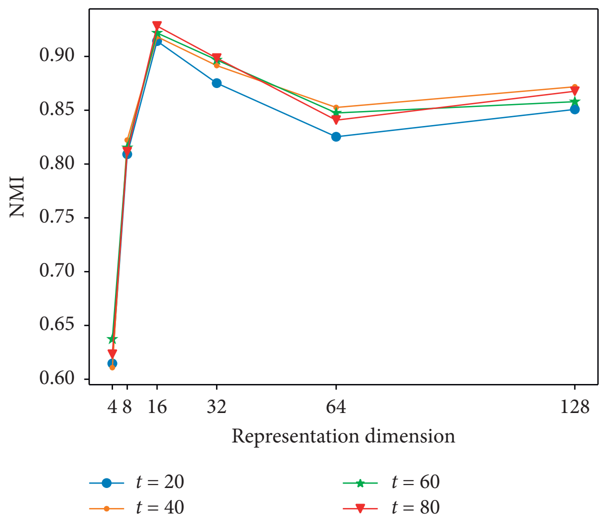
(c)



(d)

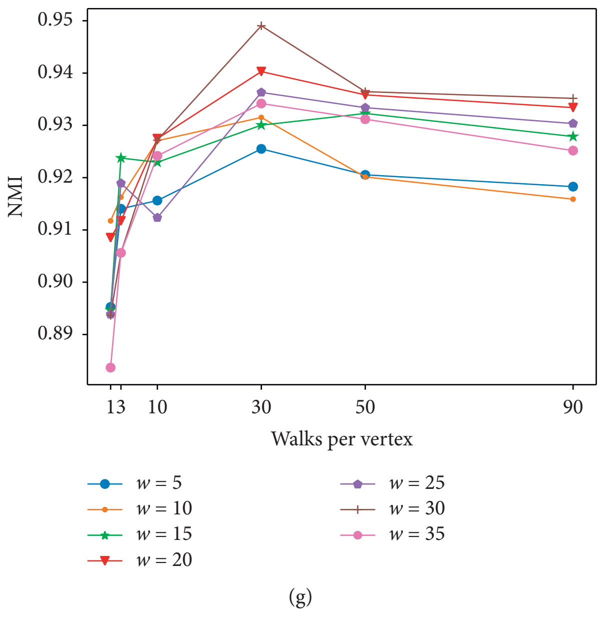


(e)



(f)

FIGURE 5: Continued.



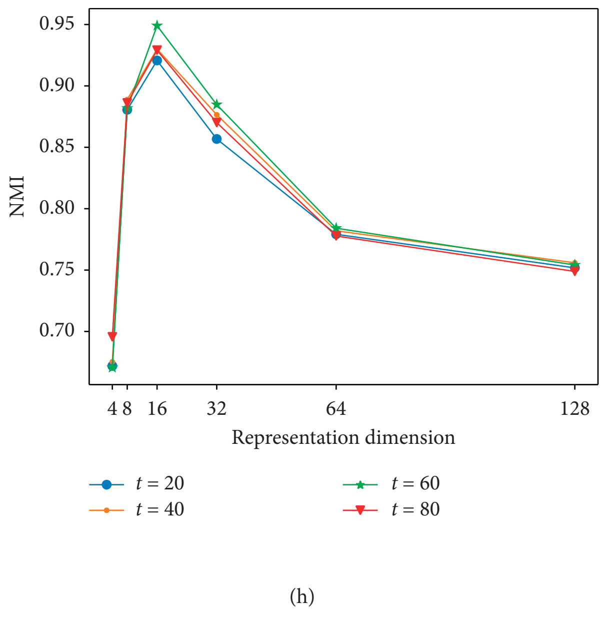


FIGURE 5: NMI for different algorithms of overlapping LFR benchmark networks. (a) and (b) correspond to LFR1 in Table 2; (c) and (d) LFR2; (e) and (f) LFR3; (g) and (h) LFR4.

TABLE 3: Comparison in terms of NMI on LFR benchmark networks.

Algorithm	LFR1	LFR2	LFR3	LFR4
GN	**1.0**	**1.0**	—	—
Louvain	0.983	**1.0**	0.914	0.964
Ours	0.986	**1.0**	**0.942**	**0.970**

Note. the entries corresponding to the best detection result are emboldened; "—" indicates that there are no running results.

running the network with 5000 nodes on our equipment 3 days, only more than 1000 edges were traversed, while the total number of edges of the network is close to 50000. The performance of the GN algorithm is indeed very good, but the time complexity is too high, $O(m^2n)$, where m is the number of edges and n is the number of nodes. This suggests that GN is not suitable for large-scale networks.

5.3.1. Time Complexity Analysis. We conducted experiments on the LFR dataset. The time spent on different datasets is shown in Figure 6, which is within acceptable limits. We set the network size N ranges from 1000 to 5000, the interval is 1000, minimum for the community sizes is 10, maximum for the community sizes is 50, and the exponent of nodes degree distribution t1 and the exponent of cluster sizes t2 equals to 2 and 1, and the mixed parameter mu is 0.2.

The change of the running time is related not only to the number of nodes but also to the walks per vertex and the dimension of the low-dimensional network. We can see the running time varies greatly from 2000 nodes to 3000 nodes, which is because at 2000 nodes, the walks are 10 and the low-dimensional space dimension is 8, which are optimal parameter. However, as the number of nodes increases, at 3000 nodes, the two parameters are 30 and 16, respectively.

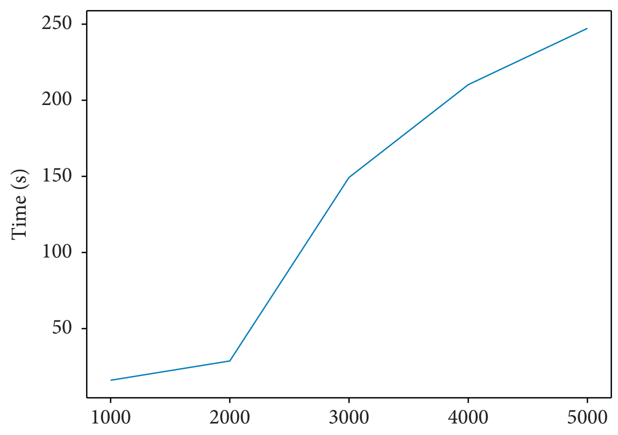


FIGURE 6: CD_DVG algorithm runtime on LFR benchmark ($N = 1000$–5000, $k = 20$, max$k = 50$, mu $= 0.2$, min$C = 10$, and max$C = 50$).

5.4. Instance Analysis. The DBLP dataset used in this paper has relatively many vertices. It is said that the vector has a large dimension and cannot visualize the clustering effect intuitively. Therefore, this experiment extracts a small dataset from DBLP for vector representation. There are 272 vertices in the dataset, which can be partitioned into two communities. In the experiment, we represent it as a two-dimensional

Table 4: Cluster label of GMM and VBGMM.

Model	Cluster label

GMM	0 1 0 3 0 3 0 0 1 0 1 0 2 1 2 3 1 2 1 2 1 1 0 4 2 0 1 2 0 2 2 2 4 0 4 1 1 2 1 0 2 1 0 1 2 0 4 3 2 1 2 0 1 2 1 2 4 1 2 2 3 0 1 2 1 0 2 2 3 2 0 1 2 4 3 2 3 2
	0 0 2 2 4 3 4 0 2 2 3 0 3 0 1 2 1 2 0 0 1 2 3 0 3 2 0 1 2 1 0 0 2 3 0 2 1 2 3 0 1 0 3 4 2 1 0 0 1 2 3 0 1 0 3 0 1 2 1 0 1 0 2 3 2 2 2 1 2 1 0 1 2 2 3 2 4
	4 2 0 1 0 3 0 1 0 4 2 3 0 1 0 1 3 2 4 0 2 2 3 0 2 1 2 0 0 1 2 0 1 2 3 2 1 2 0 2 2 0 2 3 2 3 2 0 1 2 1 2 0 1 2 3 2 1 0 4 2 3 0 1 2 1 0 1 2 2 2 2 2 4 2 4 3
	0 3 0 1 1 2 0 3 2 3 0 4 0 0 3 2 3 2 3 2 4 2 0 0 4 2 1 2 2 0 1 0 1 3 2 0 3 0 1 2

VBGMM	1 0 1 0 1 0 1 1 0 1 0 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 0 1 1 0 1 0 1 1 1 0 1 0 1 1 0 1 0 1 1 0 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 0 1 0 1
	1 1 1 1 1 0 1 1 1 1 0 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 0 1 0 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 1 1 0 1 0 1 0 1 1 0 1 1 1
	1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 1 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0 1 0
	1 0 0 1 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1

vector space, and then cluster the sample vertices by GMM and VBGMM, respectively. For comparison, in both of the two models, we set the number k of the initial clusters to be 5. In Table 4, we present the cluster labels corresponding to the sample data in the two models.

Figure 7 shows is a cluster map drawn based on the model prediction results.

Table 4 shows the clusters' labels of the sample data using the two models, respectively, and the clustering result of the sample data is shown in Figure 7. It can be observed in the table and the figure that GMM partitions the sample data into five categories (0, 1, 2, 3, and 4), while the VBGMM partitions five categories (0 and 1). The number of clusters of VBGMM is exactly equal to the real number of communities. It can be seen that the GMM can only partition the samples based on the given parameter k, while the VBGMM can cluster according to the number of actual communities in the sample data. That is the advantage of choosing the VBGMM model for clustering.

In Section 5.2, we have analysed the effect of the main parameters on the experimental results of the model. In the following of this section, we will analyze the community detection results on the DBLP dataset. The experiment is performed according to the above parameter settings. First, the vertices in the network are represented as low-dimensional vectors. Then, the low-dimensional vectors of vertices are clustered to get the cluster label of each vertex in the network. In this result, we consider each cluster as a community. Finally, based on the clustering results, we partition the network into 11 communities. Figure 8 shows the distribution of the number of vertices in each community.

Figure 9 shows a two-dimensional space community figure drawn according to the distribution of vertices in the network.

In Figure 8, we can get the distribution of vertices in the whole network: Communities 3, 4, and 9 have relatively many vertices and are called large communities. We suspect that these large communities may be core communities. Among these communities, community 4 is the largest, with more than 7,000 vertices, accounting for 25% of the total number of vertices in the network. Communities 2, 5, and 11 have the fewest number of vertices, and these communities are small. The size of communities 1, 6, 7, 8, and 10 is relatively moderate.

The size of each community in Figure 9 is clear. There are 11 colours in the picture, each of which represents a

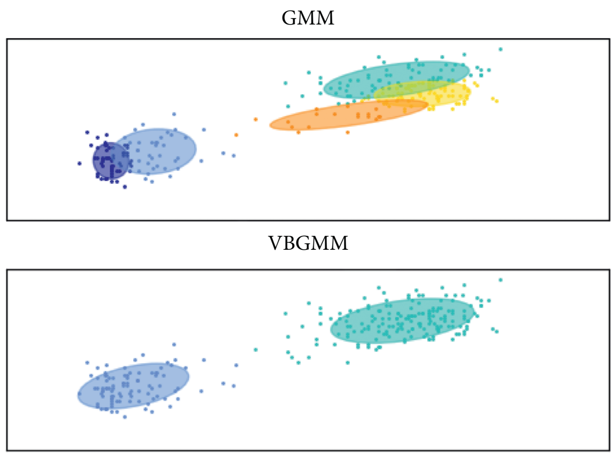


Figure 7: The clustering map of the sample data.

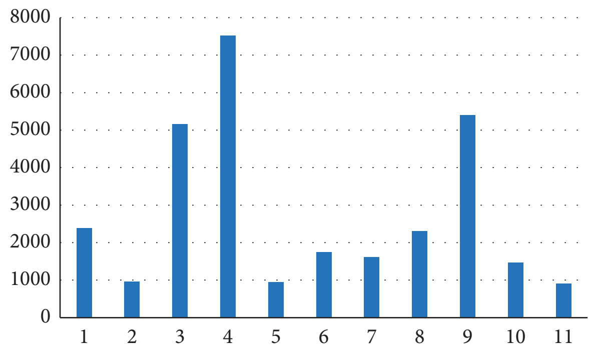


Figure 8: Distribution of the number of vertices in the community.

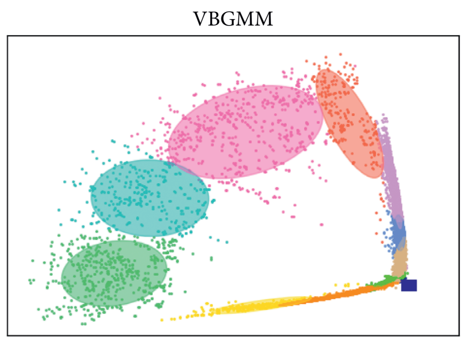


Figure 9: The distribution of vertices in the network.

Table 5: The title information of community vertex.

Community	Title of articles
C1	Document clustering and cluster topic extraction in multilingual corpora
C2	PBIR, perception-based image retrieval
C3	A parallel learning algorithm for text classification
C4	Optimal algorithms for finding user access sessions from very large web logs
C5	Recognizing multitasked activities from video using stochastic context-free grammar
C6	Feature selection for activity recognition in multirobot domains
C7	Analyzing the performance of pattern database heuristics
C8	Mining the web for answers to natural language questions
C9	UPML: a framework for knowledge system reuse
C10	Query optimization in XML-based information integration
C11	Real-time semiautomatic segmentation using a Bayesian network

community. In Table 5, we list all 11 communities and the title of articles in each community.

In Table 5, we analyze the internal characteristics of the community. One representative vertex (articles) title is selected from each community, and the category information of the community is known by analyzing the title theme. It can be seen from the table that community C1 mainly studies clustering algorithms; community C2 is mainly about image retrieval; community C3 is the study of classification algorithms; C4 is mainly about optimization algorithm; C5 mainly involves videos and images; the research content of community C6 includes feature selection, feature vector, and other issues; community C7 mainly involves the performance and management of database; community C8 mainly involves the collection and analysis of web-based data; community C9 mainly involves knowledge system and reasoning system; community C10 mainly involves various query problems in the database field; community C11 mainly studies Bayesian networks.

6. Conclusions

We propose a topology-based community detection method for large-scale networks, which applies the network representation learning and clustering algorithms. And we choose this method to mine the community structure on the DBLP dataset. The experimental results demonstrate that our method improves the accuracy of community detection. Through further analysis of the excavated community structure, the organizational characteristics within the community are better revealed. Using feature information in the directed graph and weight information of the edges to discover community is our future research direction.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (no.U1736105 and 61672297).

References

[1] S. Rani and M. Mehrotra, "Community detection in social networks: literature review," *Journal of Information & Knowledge Management*, vol. 18, no. 2, pp. 1–84, 2019.

[2] J. J. Choong, X. Liu, and T. Murata, "Variational approach for learning community structures," *Complexity*, vol. 2018, Article ID 4867304, 13 pages, 2018.

[3] B. Al-Otaibi, N. Al-Nabhan, Y. Tian et al., "Privacy-preserving vehicular rogue node detection scheme for fog computing," *Sensors*, vol. 19, no. 4, p. 965, 2019.

[4] B. Cai, L. Zeng, and Y. Wang, "Community detection method based on node density, degree centrality, and *K*-means clustering in complex network," *Entropy*, vol. 21, no. 12, p. 1145, 2019.

[5] S. Li, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3–5, pp. 75–174, 2010.

[6] V. Emilsson, M. Ilkov, J. R. Lamb et al., "Co-regulatory networks of human serum proteins link genetics to disease," *Science*, vol. 361, no. 6404, pp. 769–773, 2018.

[7] Y. Finkel, I. Koc, and I. Babaoglu, "Community detection from biological and social networks: a comparative analysis of metaheuristic algorithms," *Applied Soft Computing*, vol. 50, pp. 194–211, 2017.

[8] B. W. Kodaz and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, 1970.

[9] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 98, pp. 298–305, 1973.

[10] R. Shang, J. Bai, and L. Jiao, "Community detection based on modularity and an improved genetic algorithm," *Physica A: Statistical Mechanics and Its Applications*, vol. 392, no. 5, pp. 1215–1231, 2013.

[11] M. D. Jin and J. Biamonte, "Spectral entropies as information-theoretic tools for complex network comparison," *Physical Review X*, vol. 6, no. 4, Article ID 41062, 2016.

[12] N. Meghanathan, "Spectral radius as a measure of variation in node degree for complex network graphs," in *Proceedings of the 7th International Conference on U- and E- Service*, pp. 30–33, Hainan, China, December 2014.

[13] P. Pons and M. Latapy, "Computing communities in large networks using random walks," *Journal of Graph Algorithms and Applications*, vol. 10, no. 2, pp. 191–218, 2006.

[14] M. E. J. Newman and G. Reinert, "Estimating the number of communities in a network," *Physical Review Letters*, vol. 117, no. 7, Article ID 78301, 2016.

[15] A. Barrat, M. Barthelemy, and R. Pastor-Satorras, "The architecture of complex weighted networks," *Proceedings of the National Academy of Sciences*, vol. 101, no. 11, pp. 3747–3752, 2004.

[16] Z. Vespignani and G. Montana, "Multi-scale community detection in temporal networks using spectral graph wavelets," in *Proceedings of the International Workshop on Personal Analytics & Privacy*, pp. 139–154, Article ID 10708, Dublin, Ireland, 2017.

[17] H. Lu, X. Sang, and Q. Zhao, "Community detection algorithm based on nonnegative matrix factorization and improved density peak clustering," *IEEE Access*, vol. 8, pp. 5749–5759, 2020.

[18] L. Lu, P. Elinas, H. Nguyen et al., "Variational spectral graph convolutional networks," *Clinical Orthopaedics and Related Research*, vol. 1906, Article ID 1852, 2019.

[19] P. Cui, X. Wang, and J. Pei, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2019.

[20] L. Zhu, T. Vojković, and D. Vukicevic, "Community structure in networks: girvan-newman algorithm improvement," in *Proceedings of the 37th International Convention on Information and Communication Technology*, pp. 997–1002, Paris, France, 2014.

[21] L. Katzir and S. J. Hardiman, "Estimating clustering coefficients and size of social networks via random walk," *ACM Transactions on the Web*, vol. 9, no. 4, 2015.

[22] Z. Pan, X. Yi, Y. Zhang, F. L. Wang, and S. Kwong, "Frame-level bit allocation optimization based on," *ACM Transactions on Multimedia Computing Communications and Applications*, vol. 16, no. 1, pp. 1–23, 2020.

[23] Y. Yuan, M. M. Kaleemullah, M. A. Rodhaan et al., "A privacy preserving location service for cloud-of-things system," *Journal of Parallel and Distributed Computing*, vol. 123, pp. 215–222, 2019.

[24] Z. Q. Pan, C. N. Yang, V. S. Sheng et al., "Machine learning for wireless multimedia data security," *Security and Communication Networks*, vol. 2019, Article ID 7682306, 2 pages, 2019.

[25] T. H. Ma, H. Rong, Y. S. Hao et al., "A novel sentiment polarity detection framework for Chinese," *IEEE Transactions on Affective Computing*, vol. 1, 2019.

[26] J. Tang, M. Qu, M. Wang et al., "Line: large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077, Florence, Italy, 2015.

[27] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, New York, NY, USA, 2014.

[28] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of 22nd ACM SIGKDD International Conference*, pp. 1225–1234, San Francisco, CA, USA, 2016.

[29] X. Wang, P. Cui, J. Wang et al., "Community preserving network embedding," in *Proceedings of the 31th AAAI Conference on Artificial Intelligence*, pp. 203–209, San Francisco, CA, USA, 2017.

[30] Y. F. Chen, L. Wang, D. H. Qi et al., "Community detection based on deepwalk in large scale networks," *Communications in Computer and Information Science (CCIS)*, vol. 1210, pp. 568–583, 2020.

[31] T. Mikolov, K. Chen, G. Corrado et al., "Efficient estimation of word representations in vector space," *Clinical Orthopaedics and Related Research*, vol. 3781, 2013.

[32] J. Shi, Q. He, and Z. Wang, "GMM clustering-based decision trees considering fault rate and cluster validity for analog circuit fault diagnosis," *IEEE Access*, vol. 7, pp. 140637–140650, 2019.

[33] J. A. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," *International Computer Science Institute*, vol. 4, no. 510, pp. 1–126, 1998.

[34] D. M. Blei, A. Kucukelbir, and J. D. Mcauliffe, "Variational inference: a review for statisticians," *Journal of the American Statistical Association*, vol. 1601, Article ID 00670, 2016.