Chaos, Solitons and Fractals 189 (2024) 115630

Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

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

Community detection in directed networks based on network embeddings

Guihai Yu [a,b,*], Yang Jiao [a], Matthias Dehmer [c,e,d,f,**], Frank Emmert-Streib [g]

[a] Guizhou University of Finance and Economics, College of Big Data Statistics, Guiyang, Guizhou, 550025, China
[b] Guangxi Key Laboratory of Big Data in Finance and Economics, Guangxi University of Finance and Economics, Nanning, 530003, China
[c] Swiss Distance University of Applied Science, Department of Computer Science, 3900 Brig, Switzerland
[d] College of Artificial Intelligence, Nankai University, Tianjin 300350, China
[e] Department of Biomedical Computer Science and Mechatronics, Tyrolean Private University UMIT TIROL, A-6060 Hall in Tyrol, Austria
[f] School of Science, Xi'an Technological University, Xi'an, Shaanxi 710021, China
[g] Predictive Society and Data Analytic Lab, Faculty of Information Technology and Communication Sciences, Tampere University, Tampere 33100, Finland

ARTICLE INFO

Keywords:
Graphs
Directed networks
Network measures
Community structures
Community measures
Community optimization

ABSTRACT

In real-world scenarios, many systems can be represented using directed networks. Community detection is a foundational task in the study of complex networks, providing a method for researching and understanding the topological structure, physical significance, and functional behavior of networks. By utilizing network embedding techniques, we can effectively convert network structure and additional information into node vector representations while preserving the original network structure and properties, solving the problem of insufficient network representations. Compared with undirected networks, directed networks are more complex. When conducting community detection on directed networks, the biggest challenge is how to combine the directional and asymmetric characteristics of edges. This article combines network embedding with community detection, utilizing the cosine similarity between node embedding vectors, and combining the ComDBNSQ algorithm to achieve non overlapping community partitioning of directed networks. To evaluate the effectiveness of the algorithm, we conduct experiments using both artificial and real data sets. The numerical results indicate that the algorithm outperforms the comparison algorithms (Girvan–Newman algorithm and Label Propagation algorithm) in terms of modularity, and can perform high-quality directed network community detection.

1. Introduction

Complex networks [1] are mathematical models of real-life systems representing e.g., biological neural networks, social networks, and communication networks. In these systems, individual entities are depicted as nodes, and their interactions are symbolized by edges [2]. The concept of communities has been introduced by Girvan and Newman [3] in 2002, where communities exhibit the characteristic of being densely connected internally and sparsely connected externally [4]. Within the communities, those nodes are connected which have the same *type*, e.g., similar degree or similar value measures by a vertex complexity measure. Consequently, community detection plays an important role in analyzing network structure [5].

Community detection, fundamentally, involves segmenting a network into separate communities according to node characteristics [6]. Traditional community detection methods, such as hierarchical clustering [7], optimization methods [8], random block models [9], graph embedding [10], have received widespread attention from scholars. Most existing methods focus on detecting communities in undirected networks, where the similarity between nodes is measured by node features and whether they are connected [11]. The directionality of edges is the core of distinguishing directed and undirected networks, and the directionality of the edges often implies more complex community types [12]. In real-world scenarios, the graphs are directed in the sense that there is directionality on the edges, making the semantics of the edges non symmetric as the source node transmits some property to the target one but not vice versa. For example, in the blog reader network, there are two types of bloggers: "writers" who write influential blogs that are read by many people; "readers" read a lot, but rarely write anything for others to read. Obviously, to

* Corresponding author at: Guizhou University of Finance and Economics, College of Big Data Statistics, Guiyang, Guizhou, 550025, China.
** Corresponding author at: Department of Biomedical Computer Science and Mechatronics, Tyrolean Private University UMIT TIROL, A-6060 Hall in Tyrol, Austria.

E-mail addresses: yuguihai@mail.gufe.edu.cn (G. Yu), jiaoyang@mail.gufe.edu.cn (Y. Jiao), matthias.dehmer@umit.at (M. Dehmer), frank.emmert-streib@tuni.fi (F. Emmert-Streib).

https://doi.org/10.1016/j.chaos.2024.115630
Received 11 April 2024; Received in revised form 26 September 2024; Accepted 4 October 2024
Available online 17 October 2024

describe these two types of bloggers, it is important to examine the incoming and outgoing links of the network [13]. Community detection in directed networks is significantly applicable across various domains. Yet, despite its relevance, this area has been relatively overlooked by researchers. Even though many networks are inherently directed, the prevalent approach in community detection often disregards the directional aspect of edges. In other words, the directed network can be transformed into an undirected network (assuming edge symmetry), and then algorithms for discovering the undirected network community can be applied. However, in many cases, this simple technique is not satisfactory as it cannot capture the asymmetric relationships hidden at the edges of directed networks. Compared to undirected networks, directed networks are more complex because they involve two types of information, namely initiating or receiving links, referencing others, or being referenced, which cannot be captured by undirected network community discovery algorithms. New community detection methods for directed networks are needed to identify this asymmetric characteristic. For example, the characteristic of a directed graph is an asymmetric matrix (adjacency matrix, Laplacian matrix, etc.), making spectral analysis much more complex. Only a few methods can easily extend from undirected to directed situations. Therefore, conducting community detection on directed networks is a challenging task, for example, how to extend some concepts built on undirected graphs to directed graphs? How to extend existing theoretical tools to directed networks? Of course, the main challenge is to incorporate the directionality of edges in the community detection process.

In the field of large-scale network analysis, most methods mainly rely on edge topology information (such as edge compactness), which may lead to sparsity issues and insufficient network representation [14,15]. To address these issues, many researchers have begun to apply network embedding techniques [16] to community discovery in complex networks, aiming to generate low-dimensional dense vectors that retain the original network structure and attribute information [17–19]. A variety of network embedding methods have been developed, including those based on matrix factorization, random walks, autoencoders, and graph neural networks, which have significantly improved the performance and efficiency of community detection algorithms [20,21].

In response to the insufficient utilization of the directionality of edges in existing research on community detection in directed networks, this paper proposes a new algorithm for detecting non-over lapping communities in directed networks based on network embedding and visualization, using the ComDBNSQ algorithm proposed by Bilal and Abdelouahab [22]. The algorithm starts by utilizing network embedding to map the high-dimensional network structure information into a lower-dimensional vector space, allowing to learn the embedding vectors. By utilizing the ComDBNSQ algorithm, graph connections are then removed based on cosine similarity measure. Next, the algorithm starts an iterative process to merge communities that exhibit similarity, and to optimize modularity. The methods stops if no further improvement in modularity is detected, thus producing the final partitioning of communities.

To validate the performance of our method, we perform numerical experiments on both artificial and real networks, and comparisons are made with four existing community detection methods. Additionally, the stability and performance of the method are thoroughly analyzed, providing further evidence of its certain effectiveness, as demonstrated in this paper.

2. Theoretical foundation

Given a directed network $G(V,E)$, where V represents the set of nodes in the network and $E \subseteq V \times V$ represents the set of edges, n denotes the number of nodes, specifically $n = |V|$, and m represents the number of edges, $m = |E|$. The network structure is represented by an adjacency matrix A, in which the element a_{ij} equals 1 if there is a directed edge from node i to node j, and 0 otherwise.

Community detection [6] is the process of partitioning nodes into one or more communities. This partition is denoted by $C = \{C_1, C_2, \dots, C_k\}$, where it is essential that $C_1 \cup \cdots \cup C_k = V$. The essence of detecting communities lies in generating dense subgraphs (within the community), while the connections between communities are relatively sparse. If $\forall i,j \in [1,k]$, $C_i \cap C_j = \emptyset$, then the partition C is referred to as a non-overlapping community structure [23]. The focus of this paper lies on non-overlapping communities.

Community detection has gained prominence over the past two decades. The classic Girvan–Newman (GN) algorithm identifies communities by removing edges with high betweenness centrality [24]. The concept of modularity later emerged, enabling the assessment of community structures without prior knowledge of community distributions [25], thus advancing modularity-based community detection algorithms. The Fast Newman (FN) algorithm [26] commonly utilizes the modularity function [27] as an evaluation criterion for community partitioning, employing a greedy strategy to iteratively merge communities and top optimize modularity. However, due to its high computational complexity, Clauset et al. [28] proposed the CNM algorithm, i.e., an agglomerative approach based on the greedy method. It generates an increment matrix ΔQ to efficiently discover communities with maximum modularity. Notably, the algorithm only update the elements of the increment matrix during its implementation, resulting in a time complexity close to linear. In addition, Raghavan et al. [29] introduced the Label Propagation Algorithm (LPA), which identifies all communities through local label propagation. Although the LPA algorithm has a lower time complexity, but its random behavior and propagation mechanism may lead to the oversight of smaller communities and less stable results.

Network embedding [30] is a method that maps graph data to a low-dimensional vector space, transforming intricate graph structures into continuous vector representations. This enables nodes with similar characteristics to have analogous vector representations. This process preserves the topological structure of the network, including proximity relationships among nodes and edges, as well as community information, within the reduced-dimensional space. Locally Linear Embedding (LLE) [31] employs matrix factorization to map nodes as linear combinations of neighbors, preserving local relationships while reducing complexity with sparse matrices. Laplacian Eigenmaps (LE) [32] use the adjacency matrix to embed nodes with local structural details, reflecting node similarities and connections. Inspired by word embeddings, DeepWalk [33] creates node sequences via random walks and trains representations with the Skip-Gram model. Large-scale Information Network Embedding (LINE) [34] enhances first-order neighborhood sparsity by considering second-order neighborhoods, leading to more meaningful node relationship analysis.

The quality of a community is determined by the level of cohesion among its members and the separation between different communities. Modularity, proposed by Newman [35], is a measure for evaluating community quality. It measures the dissimilarity between a real network and its corresponding null (random) model. A higher modularity value indicates a more reasonable community structure, indicating a better partitioning of communities. The closer the modularity value is to 1, the more clear/strong is the community structure. In the case of directed networks, the formula for calculating modularity [36] is as follows:

$$Q_d = \frac{1}{m} \sum_{i,j} (a_{ij} - \frac{k_i^{out} k_j^{in}}{m}) \delta(c_i, c_j) \tag{1}$$

Here, m represents the number of edges in the network. $A = (a_{ij})_{n \times n}$ is the adjacency matrix of the network, where a_{ij} is an element of the adjacency matrix. k_i^{out} represents the out-degree of node i, k_j^{in} represents the in-degree of node j. c_i denotes the community to which node i belongs, c_j denotes the community to which node j belongs. The formula for $\delta(c_i, c_j)$ is as follows:

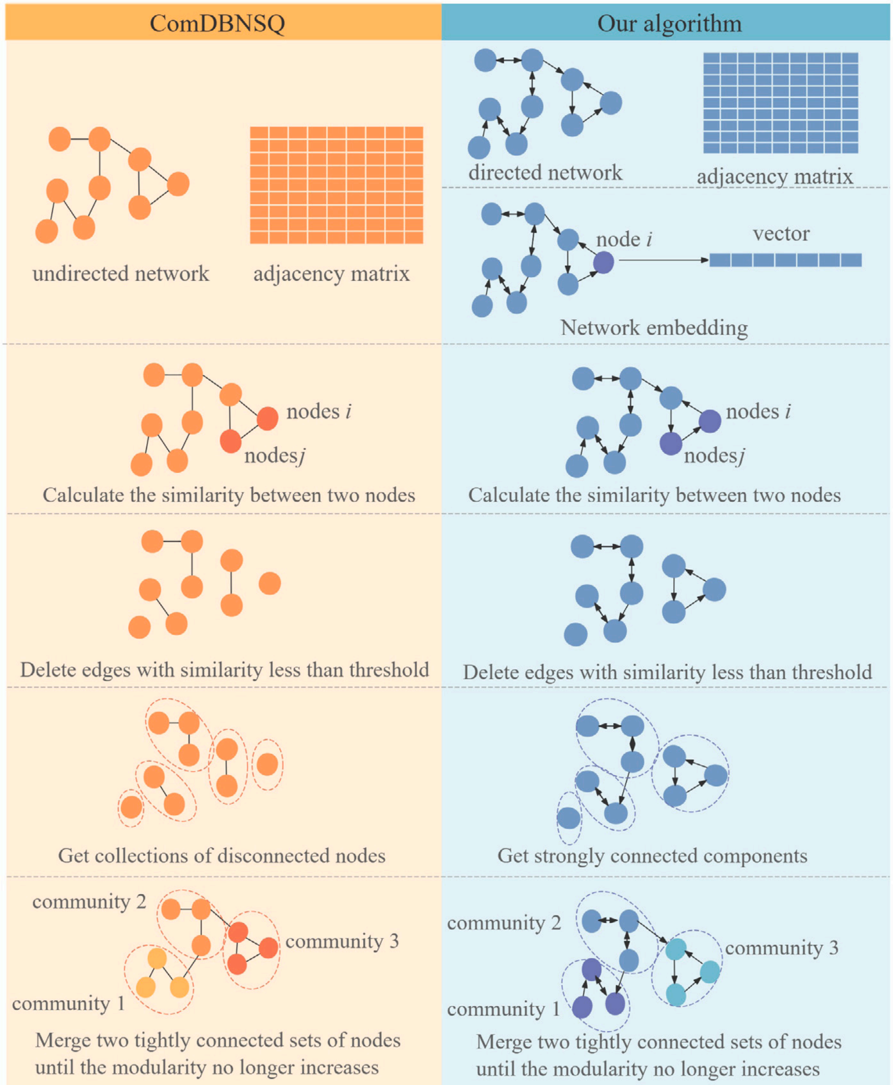

Fig. 1. Flow chart of ComDBNSQ and our algorithm.

$$\delta(c_i, c_j) = \begin{cases} 1, & c_i = c_j \\ 0, & c_i \neq c_j \end{cases} \tag{2}$$

Among them, the meaning of $\delta(c_i, c_j)$ is as follows: If node i and node j belong to the same community, $\delta(c_i, c_j) = 1$; otherwise, $\delta(c_i, c_j) = 0$.

3. Novel algorithm

In this section, this paper introduces an improved ComDBNSQ algorithm based on network embedding techniques, namely LLE-based, DeepWalk-based, LE-based, and LINE-based, aimed at applying it to community detection in directed networks, see Fig. 1.

The original ComDBNSQ algorithm [22] is primarily designed for undirected networks. Its core concept is to measure the similarity between nodes based on the similarity definition, assign weights to edges, and remove edges with weights below a preset threshold. In undirected networks, the algorithm identifies small, disconnected sets of nodes and merges closely connected communities iteratively with the goal of optimizing modularity, until no further improvement in modularity is observed, ultimately achieving the community partitioning result.

The algorithm presented in this paper is tailored for directed networks and begins by obtaining the embedded vectors of nodes through network embedding techniques. This process not only generates low-dimensional and dense vectors but also effectively addresses issues such as high dimensionality, sparsity, noise, and information redundancy while preserving the original network structure and attribute information. Subsequently, the algorithm uses cosine similarity to measure the similarity between nodes and assigns appropriate weights to edges, removing those with weights below the threshold. Building on this, the algorithm further identifies the strongly connected components in the directed graph. Finally, adhering to the principle of modularity optimization, it iteratively merges closely connected communities until no further increase in modularity is detected, thereby obtaining the final community partition.

Now, let us explain the two major steps of our algorithm:

- Step 1: Learn node embeddings using network embedding methods in directed networks, as illustrated in Fig. 1. This paper employs four network embedding methods: LLE [31], DeepWalk [33], LE [32], and LINE [34]. Among them, LLE is a matrix factorization-based method that represents each node as a linear combination of its neighboring nodes, constructing a mapping that preserves the neighborhood structure. During the dimensionality reduction process, LLE can preserve the local characteristics of samples and appropriately reduce the computational complexity through sparse matrix factorization. The DeepWalk algorithm utilizes a random walk approach to generate node sequences, treating them as textual data. Subsequently, the Word2Vec model is employed to derive embedded representations of the nodes, resulting in high efficiency and accuracy when processing large-scale graph data. Similarly to LLE, the Laplacian Eigenmaps (LE) method uses the adjacency matrix to construct embeddings that incorporate local structural information. LINE, a large-scale information network embedding method is based on rich second-order neighborhoods to compensate for the sparsity of first-order neighbors. In the process of network embedding, parameters such as the dimensionality of the network embedding, the number of neighboring nodes, and the length of random walks need to be set by oneself. To optimize the trade-off between performance and fine-tuning of parameters, we use exhaustive search to set these parameters[1]

Furthermore, we apply the algorithm based on the LLE network embedding as LLE-based, the algorithm based on DeepWalk network embedding as DeepWalk-based, the algorithm based on LE network embedding as LE-based, and the algorithm based on LINE network embedding as LINE-based.

- Step 2: Building upon the initial step of the ComDBNSQ algorithm introduced by Bilal and Abdelouahab [22], the graph connections are eliminated and partitioned into several relatively small disjoint sets of nodes. In this paper, cosine similarity is employed to quantify the similarity between nodes. Cosine similarity is defined as follows:

$$S(A,B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{t=1}^{k} A_t B_t}{\sqrt{\sum_{t=1}^{k} A_t^2}\sqrt{\sum_{t=1}^{k} B_t^2}}, \quad S(A,B) \in [-1,1] \tag{3}$$

The vectors A and B represent the embedding vectors of the corresponding nodes, and k is the dimension of the embedding vectors. After obtaining the embedding vectors, the cosine similarity between them is computed, and the corresponding edge weights are assigned. A threshold θ, where $\theta \in [-1,1]$, is set. Subsequently, we filter the edges in the network by removing those with weights below a predetermined threshold to obtain the strongly connected components of the network. We place the strongly connected components that consist of a single node into the set C_r, and denote the other strongly connected components that include multiple nodes as C_i, where $C_i \in (C_1, C_2, \ldots, C_p)$.

Then, each node i in C_r is added to the community C_i that maximizes the value of the ratio F_c. The expression for F_c is as follows:

$$F_C = \frac{|\Gamma(i) \cap C_j|}{|C_j|}, \quad \begin{cases} i \in C_r \\ C_j \in (C_1, C_2, \ldots, C_p) \end{cases} \tag{4}$$

Here, $\Gamma(i)$ represents the set of neighboring nodes of node i, and $|C_j|$ indicates the size of community C_j, which corresponds to the number of nodes within that community. Following this step, a set comprising p partitions $\pi(\pi = C_1, C_2, \ldots, C_p)$ is obtained, where nodes within each partition exhibit close connections and similarities to one another.

- Step 3: Expanding upon the second step of the ComDBNSQ algorithm introduced by Bilal and Abdelouahab [22], the ultimate community structure is established through iterative procedures. The p sets of nodes acquired in the second step are merged iteratively until the modularity cannot be further enhanced. Notably, the precise number of communities k does not need to be pre-determined. Initially, the matrix $X_{p\times p}$ is defined by,

[1] Due to the extensive volume of data, the article does not include all the exhaustive experimental results. For a complete set of detailed experimental outcomes, please reach out to the corresponding author.

Table 1
The time complexity of network embedding algorithms and comparison algorithms in this article.

Algorithm	Time complexity
LLE	$O(n^2)$
$DeepWalk$	$O(n) + O(m) + O(n \log n)$
LE	$O(n)$
$LINE$	$O(dkm)$
GN	$O(nm^2)$
FN	$O(n(m+n))$
CNM	$Q(n \log^2 n)$
LPA	$O(m)$

$$X_{p\times p} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{p1} & x_{p2} & \cdots & x_{pp} \end{bmatrix}, \tag{5}$$

where

$$\begin{cases} x_{ii} = 0 \\ x_{ij} = \frac{number\ of\ edges\ between\ C_i\ and\ C_j}{|C_i|+|C_j|}, \quad i \neq j \end{cases}$$

Within the matrix X, the diagonal elements x_{ii} are set to 0, where x_{ij} signifies the ratio of the number of edges between community C_i and C_j to the sum of their sizes. During each iteration, the maximum value within matrix X is identified, and the corresponding communities are merged accordingly. Afterwards, the modularity of the new community partition is computed, and this procedure is repeated. We continue iterating until the modularity ceases to increase, ultimately yielding the final community partition. It is noteworthy that, due to the inherent resolution limit of modularity itself [37], this algorithm, as a modularity-based community detection method, may have certain limitations in identifying smaller-scale communities.

3.1. Algorithm complexity analysis

This paper proposes a community detection algorithm for directed networks based on network embedding, drawing inspiration from the ComDBNSQ algorithm [22]. In the forthcoming description, n represents the number of nodes in the network, m represents the number of edges, d represents the dimension of the embedding vectors, and k represents the number of negative samples. Firstly, a network embedding method is employed to learn the embedding vectors of nodes. The time complexity can be seen in Table 1. Next, the cosine similarity between the embedding vectors of nodes is computed, and a threshold is set to remove edges. This step requires a time complexity of $O(m)$. Following that, small communities with more than one node are added to the set π, and isolated nodes are added to the set C_r. This step has a time complexity of $O(n)$. Afterwards, isolated nodes are assigned to the communities where their majority of neighboring nodes belong. This operation has a time complexity of $O(n^2)$. Finally, communities are merged based on the principle of maximizing modularity until no further increase can be determined. The time complexity of this step is $O(n^2)$. In summary, the overall time complexity of this algorithm is the sum of the time complexity of the network embedding method and $O(n^2+m)$.

4. Experiments and analysis

The experimental setup involved using an Intel(R) Core(TM) i5-12500H 2.50 GHz processor with 16 GB of RAM. The operating system chosen for the experiments was Windows 11, and Python 3.9.13 was utilized as the programming language.

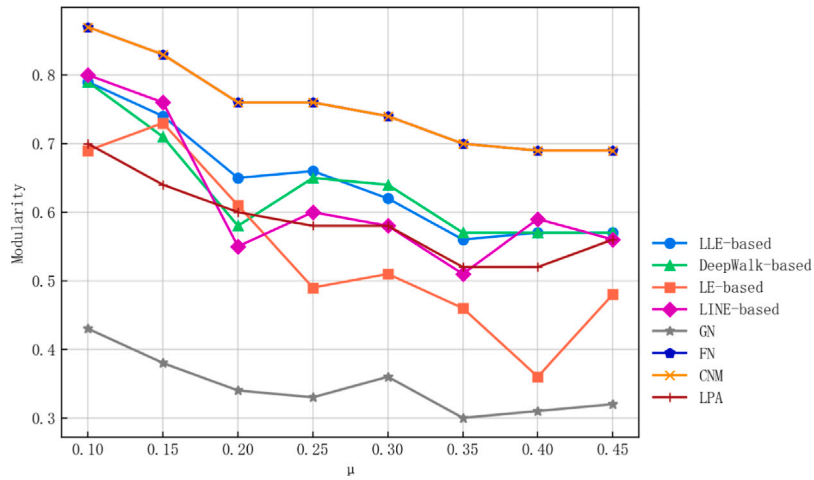

Fig. 2. Results of experiments conducted on artificial data sets.

Table 2
Directed artificial network data set information.

Data set	Nodes	Edges	k
$LFR-1$	286	500	1.75
$LFR-2$	293	500	1.71
$LFR-3$	291	500	1.72
$LFR-4$	282	500	1.77
$LFR-5$	282	500	1.71
$LFR-6$	285	500	1.75
$LFR-7$	284	500	1.76
$LFR-8$	282	500	1.77

4.1. Experimental results and analysis on artificial datasets

In this section, we evaluate the performance of the algorithm using the LFR benchmark data set [38]. The LFR network, unlike other artificial network datasets, offers adjustable parameters such as node degree and community size. This flexibility allows it to effectively capture the heterogeneity in node degrees and community size distributions, making it a better reflection of real-world networks. For this study, we generated eight artificial networks using the LFR benchmark, labeled as LFR-1 to LFR-8. These networks consisted of 300 nodes with an average node degree of 10, a maximum node degree of 20, and community sizes ranging from 2 to 10. The mixing parameter μ varied from 0.1 to 0.45. Since the generated networks were initially undirected, we converted them into directed networks. For each artificial network, we randomly selected 500 edges and constructed a directed graph. The fundamental information of the resulting directed graph is summarized in Table 2, where “Nodes” represents the number of nodes, “Edges” represents the number of edges, and “k” represents the average degree of the network.

In the case of artificial data sets, we use the modularity score Q. The algorithm proposed in this paper was compared to the classical GN, FN, CNM, and LPA algorithms. The experimental results of these algorithms on networks with different complexities are illustrated in Fig. 2.

As depicted in Fig. 2, as the value of μ ranges from 0.1 to 0.45, the artificial network structures become increasingly complex, resulting in varying degrees of decline in modularity across all algorithms. In terms of modularity as the evaluation metric, the FN and CNM algorithms achieve better community partitioning results. As the network structures become more intricate, these algorithms experience a relatively lower decrease in modularity, demonstrating more stability and obtaining the highest modularity among all community partitioning results. On the other hand, the community partitioning results obtained from the LLE-based, DeepWalk-based, LINE-based, and LPA algorithms show no significant differences in modularity or trends as the network complexity increases, suggesting that these algorithms perform similarly on networks of different complexities. The LE-based algorithm yields results with larger fluctuations, indicating relatively less stability in community partitioning as the network complexity increases. It is noteworthy that the GN algorithm exhibits minimal variation in community partitioning results as μ transitions from 0.1 to 0.45, consistently maintaining modularity within the range of [0.3,0.45], thereby demonstrating higher stability. However, compared to other algorithms, the GN algorithm yields lower modularity in community partitioning results.

4.2. Experimental results and analysis on real datasets

4.2.1. Experimental setup

In order to assess the performance of the algorithm proposed in this paper, simulations were performed on six publicly available directed network datasets (six real network data sets). The community partitioning results were then compared to those obtained from four classical algorithms (GN, FN, CNM, and LPA).

$Friendship-network-data-2013$ data set, a friendship network that records the contact and friendship relationships among high school students in Marseille, France in December 2013. $Celegans-n306$ data set, a neural network of the Caenorhabditis elegans roundworm. Two neurons are considered connected if there is at least one synaptic or gap junction connection between them. Convote data set, a voting record data set where nodes represent speakers, and directed edges indicate one speaker referencing another speaker. $Moreno-innovation$ data set, a social network data set of 241 doctors, where each node represents a doctor and edges represent relationships between doctors. Foodweb data set, a biological network from Little Rock Lake, Wisconsin. Nodes represent autotrophs, herbivores, carnivores, and decomposers, while edges represent food relationships. $Wiki-talk$ data set, a communication network from Wikipedia. Nodes represent users, and an edge from user A to user B indicates that user A wrote a message on user B’s talk page at a specific time. The above networks are all public data sets from sociopatterns.org and toreopsahl.com.

Note: For convenience in subsequent discussions, the $Friendship-network-data-2013$ data set, $Celegans-n306$ data set, Convote data set, $Moreno-innovation$ data set, Foodweb data set, and $Wiki-talk$ data set will be referred to as Data set 1, Data set 2, Data set 3, Data set 4, Data set 5, and Data set 6, respectively. This naming convention will be used when referring to these data sets later on. Table 3 provides the statistical data for these six publicly available directed networks. In the table, “Nodes” indicates the number of nodes, “Edges” represents the number of edges, “k” denotes the average degree of the network, and “Network type” specifies the network type.

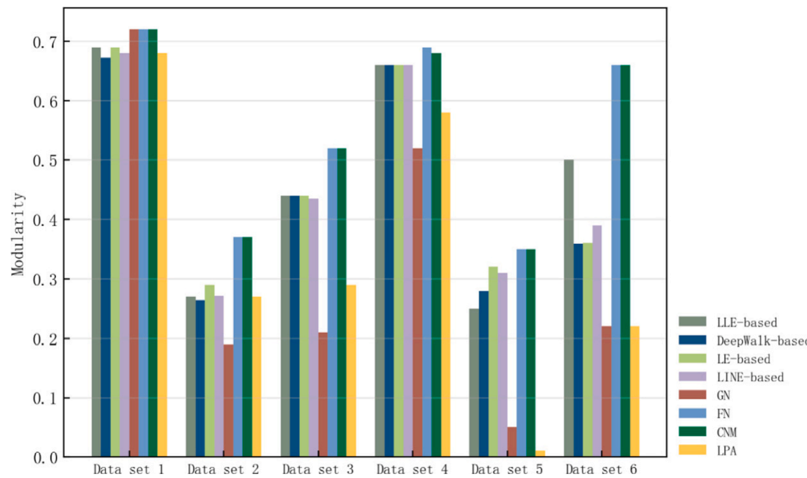

Fig. 3. Comparison of experimental results of comparison algorithms in this article and our algorithms.

Table 3
The basic information of real networks for experimentation.

Data set	Nodes	Edges	k	Network type
Data set 1	134	668	4.99	Social
Data set 2	297	2345	7.9	biological
Data set 3	219	588	2.68	Social
Data set 4	241	1098	4.56	Social
Data set 5	183	2494	13.63	biological
Data set 6	536	957	1.79	correspondence

4.2.2. Results and analysis on real datasets

To test the modularity and runtime of the algorithm proposed in this paper on real networks, the algorithm was independently executed 10 times on each data set. The average modularity and runtime are presented in Table 4, with time measured in seconds.

Firstly, during the community partitioning process on these six directed networks, the method proposed in this paper achieved modularity values within the range of [0.25, 0.69]. This indicates a relatively high quality of the communities found. Secondly, the algorithm has a runtime within the range of [0.09, 0.62], demonstrating its relatively fast computational speed. Next, a comparison will be conducted with several classical algorithms applied on real networks. Fig. 3 displays the experimental results obtained using the GN, FN, CNM, and LPA algorithms.

Based on the results, it is evident that all algorithms performed well in the community detection on Data set 1, yields to a modularity values around 0.7. Variations towards the modularity values among the algorithms are relatively small. For Data set 4, except for the GN and LPA algorithms, the remaining algorithms achieved modularity scores of 0.65 or higher, indicating that the communities have quite outcomes.

By using the remaining data sets, the FN and CNM algorithms demonstrated strong performance in community partitioning, attaining the highest modularity scores. Note that the GN and LPA algorithms produced less satisfactory results. Regarding the proposed algorithm, when compared to other network embedding-based approaches, LLE-based gives certain advantages towards community detection on Data set 6. However, the differences in the results obtained by different embedding methods are not significant in community detection based on other data sets.

4.2.3. Stability analysis of the algorithm

Due to the increasing scale of networks, the community structure becomes more intricate, and many community detection algorithms show notable instability. This implies that running the community detection algorithms on the same network can result in substantial differences in the resulting communities. To evaluate the stability of the algorithm proposed in this paper, independent experiments were conducted twenty times on each data set, and the experimental results for modularity were obtained, as illustrated in Fig. 4. The experimental results indicate that in the twenty independent experiments, LLE-based and LE-based methods demonstrated good stability in terms of modularity. However, there were fluctuations in the stability of community detection for DeepWalk-based and LINE-based algorithms. Specifically, the DeepWalk-based algorithm exhibited slight fluctuations when partitioning Data set 1, Data set 4, and Data set 5; while more pronounced fluctuations were observed on other data sets. This is mainly due to the differences in the random walk sequences in each experiment, which may fail to fully capture the global structure of the network. Moreover, the choice of parameters such as walk length, number of walks, and embedding dimensions has a significant impact on the final results, requiring careful adjustment based on the specific network. For the LINE-based algorithm, the experimental results showed better stability when partitioning Data set 4 and Data set 6. However, larger fluctuations were observed when partitioning Data set 3, and some degree of fluctuation was also observed on the other three data sets. This is primarily because for nodes with lower degrees, the LINE-based algorithm may struggle to obtain high-quality embedded representations, as these nodes have a limited number of neighbors, leading to insufficient information in community detection based on second-order similarity. At the same time, the choice of parameters significantly affects the results, necessitating optimization adjustments for specific networks.

4.2.4. Performance comparison analysis

To accurately evaluate the performance of the algorithm proposed in this paper, we conducted an exhaustive experimental study on each real directed network data sets. The experiment started with an empty network, i.e., the initial node count was zero, and at each step, we randomly selected a subset of nodes from the existing network to add, applying the algorithm proposed in this paper for community detection. Given that the selection of different nodes when random sampling may lead to variations in experimental results, and that during the process of community detection, algorithms based on DeepWalk and LINE exhibit fluctuations in terms of stability (see Section 4.2.3), the experimental results have a certain degree of randomness. Therefore, we conducted multiple experiments to obtain more reliable data. As shown in Fig. 5, through simulation experiments, we observed that the modularity index of the algorithm proposed in this paper exhibits a slight linear growth trend during the community detection process, with some fluctuations afterwards, but it eventually stabilizes. This

Table 4
The experimental results of our algorithms on real networks.

Algorithm	LLE-based		DeepWalk-based		LE-based		LINE-based	
Data set	Modularity	time	Modularity	time	Modularity	time	Modularity	time
Data set 1	0.69	0.39	0.67	0.10	0.69	0.16	0.68	0.09
Data set 2	0.27	0.62	0.26	0.37	0.29	0.53	0.27	0.35
Data set 3	0.44	0.09	0.44	0.13	0.44	0.10	0.44	0.09
Data set 4	0.66	0.10	0.66	0.17	0.66	0.28	0.66	0.16
Data set 5	0.25	0.14	0.28	0.24	0.32	0.23	0.31	0.25
Data set 6	0.53	0.30	0.36	0.28	0.36	0.27	0.39	0.32

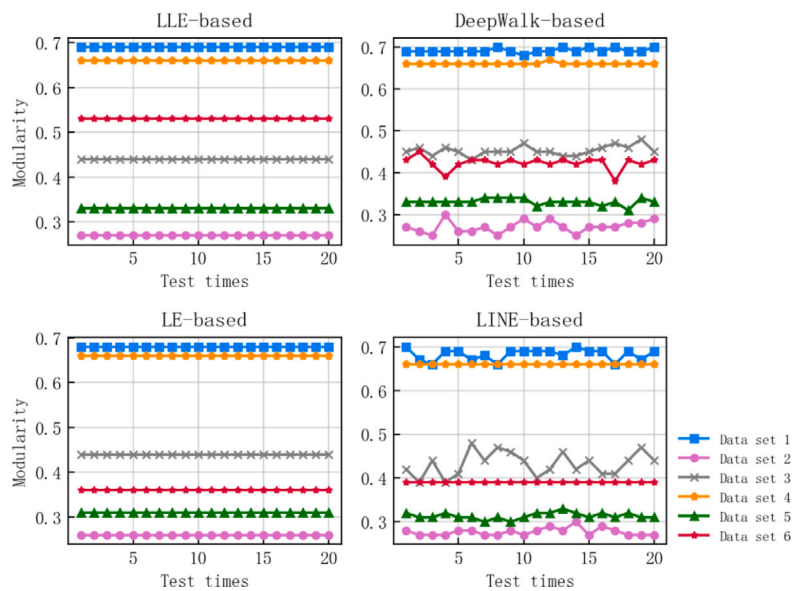

Fig. 4. Algorithm stability analysis diagram.

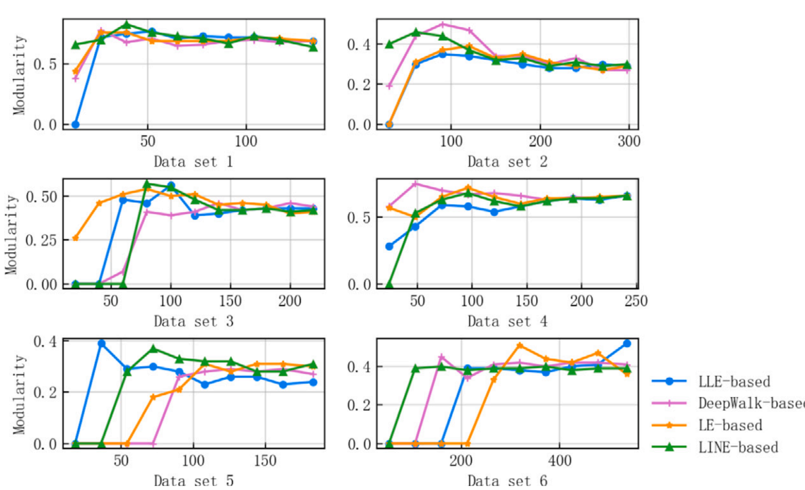

Fig. 5. Algorithm performance analysis diagram.

phenomenon indicates that as the number of network nodes increases, the modularity results obtained from community detection using the algorithm proposed in this paper gradually become stable, verifying the effectiveness and stability of the algorithm.

5. Summary and conclusion

In this paper, we developed a community detection algorithm for directed network based on network embeddings. Four network embedding methods, namely LLE, DeepWalk, LE, and LINE, are utilized to obtain embedding vectors for nodes. The cosine similarity is calculated to measure the similarity between nodes. Based on this, the algorithm removes the connections in the graph and merges communities according to the optimality principle of modularity. Subsequently, experiments were performed on eight artificial data sets and six real data sets. The proposed algorithm was compared with GN, FN, CNM, and LPA algorithms. Also, we analyzed the stability of the algorithm and compared the performance with others.

On artificial data sets with varying levels of complexity, the algorithm proposed in this paper, based on different network embeddings, consistently produced community partitioning results of comparable quality, evaluated by using the modularity score. After comparing the results, we have found that the proposed algorithm outperforms the GN and LPA algorithm in terms of modularity.

Moreover, the algorithm proposed in this paper based on network embedding has some limitations. Given the fact that our method can find communities well and accurately, the algorithm has a constrained by a resolution limit. This means that it often struggles to identify small-scale community structures when the modularity value reaches its maximum. The situation is then similar to clique search. Also, the algorithm based on network embedding requires pre-determination of certain parameter values during its usage, such as the dimensionality of network embedding, the number of neighboring nodes, the length of random walks, and the threshold for similarity. These parameters are closely tied to the quality of the network embedding vectors and the resulting community partitioning quality. Consequently, finding appropriate thresholds becomes a challenging problem that we want to tackle as future work.

CRediT authorship contribution statement

Guihai Yu: Writing – original draft. **Yang Jiao:** Writing – original draft. **Matthias Dehmer:** Writing – original draft. **Frank Emmert-Streib:** Writing – original draft.

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

Data will be made available on request.

Acknowledgments

Guihai Yu and Yang Jiao were supported by National Natural Science Foundation of China (11861019, 62363004, 12461062), Natural Science Foundation of Guizhou ([2019]1047, [2020]1Z001, [2021]5609), Guangxi Key Laboratory of Big Data in Finance and Economics (FEDOP2022A01).

References

[1] Hu Feng, Qiu Liping, Wei Shaobin, Zhou Haiyan, Bathuure Isaac Akpemah, Hu Hao. The spatiotemporal evolution of global innovation networks and the changing position of china: a social network analysis based on cooperative patents. R & D Manage 2024;54(3):574–89.

[2] Yu Zhiyong, Chen Jijie, Guo Kun, Chen Yuzhong, Xu Qian. Overlapping community detection based on influence and seeds extension. Acta Electon Sinica 2019;47(1):153.

[3] Girvan Michelle, Newman Mark EJ. Community structure in social and biological networks. Proc Natl Acad Sci 2002;99(12):7821–6.

[4] Kaur Davinder, Uslu Suleyman, Durresi Mimoza, Durresi Arjan. A geo-location and trust-based framework with community detection algorithms to filter attackers in 5 g social networks. Wirel Netw 2024;30(6):4841–9.

[5] Sun Zejun, Sun Yanan, Chang Xinfeng, Wang Qiming, Yan Xuetong, Pan Zhongqiang, et al. Community detection based on the matthew effect. Knowl-Based Syst 2020;205:106256.

[6] Huang Xinyu, Chen Dongming, Ren Tao, Wang Dongqi. A survey of community detection methods in multilayer networks. Data Min Knowl Discov 2021;35:1–45.

[7] Fortunato Santo, Hric Darko. Community detection in networks: A user guide. Phys Rep 2016;659:1–44.

[8] Li Xianghua, Qi Xin, Liu Xingjian, Gao Chao, Wang Zhen, Zhang Fan, et al. A discrete moth-flame optimization with an l_2-norm constraint for network clustering. IEEE Trans Netw Sci Eng 2022;9(3):1776–88.

[9] Yang Liang, Cao Xiaochun, He Dongxiao, Wang Chuan, Wang Xiao, Zhang Weixiong. Modularity based community detection with deep learning. In: IJCAI, vol. 16. 2016, p. 2252–8.

[10] Zhu Junyou, Wang Chunyu, Gao Chao, Zhang Fan, Wang Zhen, Li Xuelong. Community detection in graph: An embedding method. IEEE Trans Netw Sci Eng 2021;9(2):689–702.

[11] Zhang Jingnan, He Xin, Wang Junhui. Directed community detection with network embedding. J Amer Statist Assoc 2022;117(540):1809–19.

[12] Smith Natalie R, Zivich Paul N, Frerichs Leah M, Moody James, Aiello Allison E. A guide for choosing community detection algorithms in social network studies: The question alignment approach. Am J Preventive Med 2020;59(4):597–605.

[13] Jin Di, Yu Zhizhi, Jiao Pengfei, Pan Shirui, He Dongxiao, Wu Jia, et al. A survey of community detection approaches: From statistical modeling to deep learning. IEEE Trans Knowl Data Eng 2021;35(2):1149–70.

[14] Tu Cunchao, Zeng Xiangkai, Wang Hao, Zhang Zhengyan, Liu Zhiyuan, Sun Maosong, et al. A unified framework for community detection and network representation learning. IEEE Trans Knowl Data Eng 2018;31(6):1051–65.

[15] Zhang Daokun, Yin Jie, Zhu Xingquan, Zhang Chengqi. Network representation learning: A survey. IEEE Trans Big Data 2018;6(1):3–28.

[16] Goyal Palash, Ferrara Emilio. Graph embedding techniques, applications, and performance: a survey. Knowl-Based Syst 2018;151:78–94.

[17] Cui Peng, Wang Xiao, Pei Jian, Zhu Wenwu. A survey on network embedding. IEEE Trans Knowl Data Eng 2018;31(5):833–52.

[18] Li Xinshuai, Pan Limin, Zhou Yanru, Wu Zhouting, Luo Senlin. A temporal-spatial network embedding model for ict supply chain market trend forecasting. Appl Soft Comput 2022;125:109118.

[19] Wang Xiao, Bo Deyu, Shi Chuan, Fan Shaohua, Ye Yanfang, Philip SYu. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. IEEE Trans Big Data 2022;9(2):415–36.

[20] Athar Sajjad, Abbasi Rabeeh Ayaz, Saeed Zafar, Said Anwar, Razzak Imran, Salim Flora D. Asbine: Dynamic bipartite network embedding for incorporating structural and attribute information. World Wide Web 2023;26(5):3463–81.

[21] Wang Yaojing, Yao Yuan, Tong Hanghang, Xu Feng, Lu Jian. A brief review of network embedding. Big Data Mining Anal 2018;2(1):35–47.

[22] Saoud Bilal, Moussaoui Abdelouahab. Node similarity and modularity for finding communities in networks. Physica A 2018;492:1958–66.

[23] Fu Luoyi, Zhang Jiapeng, Wang Shuaiqi, Wu Xinyu, Wang Xinbing, Chen Guihai. De-anonymizing social networks with overlapping community structure. IEEE/ACM Trans Netw 2020;28(1):360–75.

[24] Kiruthika R, Vijaya MS. Community detection using girvan-newman and kernighan-lin bipartition algorithms. In: Data intelligence and cognitive informatics: proceedings of ICDICI 2021. Springer; 2022, p. 217–31.

[25] Lambiotte Renaud, Schaub Michael T. Modularity and dynamics on complex networks. Cambridge University Press; 2021.

[26] Newman Mark EJ, Girvan Michelle. Finding and evaluating community structure in networks. Phys Rev E 2004;69(2):026113.

[27] Yuan Quan, Liu Binghui. Community detection via an efficient nonconvex optimization approach based on modularity. Comput Statist Data Anal 2021;157:107163.

[28] Fortunato Santo. Community detection in graphs. Phys Rep 2010;486(3–5):75–174.

[29] Leung Ian XY, Hui Pan, Lio Pietro, Crowcroft Jon. Towards real-time community detection in large networks. Phys Rev E–Statist Nonlinear Soft Matter Phys 2009;79(6):066107.

[30] Zhang Liangpei, Luo Fulin. Review on graph learning for dimensionality reduction of hyperspectral image. Geo-spatial Inf Sci 2020;23(1):98–106.

[31] Fang Yu, Li Hao, Ma Yong, Liang Kun, Hu Yingjie, Zhang Shaojie, et al. Dimensionality reduction of hyperspectral images based on robust spatial information using locally linear embedding. IEEE Geosci Remote Sens Lett 2014;11(10):1712–6.

[32] Hou Biao, Zhang Xiangrong, Ye Qiang, Zheng Yaoguo. A novel method for hyperspectral image classification based on laplacian eigenmap pixels distribution-flow. IEEE J Sel Top Appl Earth Obs Remote Sens 2013;6(3):1602–18.

[33] Perozzi Bryan, Al-Rfou Rami, Skiena Steven. Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. 2014, p. 701–10.

[34] Tang Jian, Qu Meng, Wang Mingzhe, Zhang Ming, Yan Jun, Mei Qiaozhu. Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web. 2015, p. 1067–77.

[35] Zelditch Miriam L, Goswami Anjali. What does modularity mean? Evolut Develop 2021;23(5):377–403.

[36] Chakraborty Tanmoy, Dalmia Ayushi, Mukherjee Animesh, Ganguly Niloy. Metrics for community analysis: A survey. ACM Comput Surv 2017;50(4):1–37.

[37] Guo Jiahao, Singh Pramesh, Bassler Kevin E. Resolution limit revisited: community detection using generalized modularity density. J Phys Complexity 2023;4(2):025001.

[38] Xiao Xinjie, Li Zhiwei, Ning Wenle, Zhang Nannan, Teng Xudong. Lfr-net: Local feature residual network for single image dehazing. Array 2023;17:100278.