Hindawi
Complexity
Volume 2020, Article ID 9017239, 14 pages
https://doi.org/10.1155/2020/9017239

Research Article

A Seed-Expanding Method Based on TOPSIS for Community Detection in Complex Networks

Jianjun Cheng,[1] **Wenbo Zhang**,[1] **Haijuan Yang**,[1,2] **Xing Su**,[1] **Tao Ma**,[3] **and Xiaoyun Chen**[1]

[1]*School of Information Science and Engineering, Lanzhou University, Lanzhou, China*
[2]*Department of Electronic Information Engineering, Lanzhou Vocational Technical College, Lanzhou, China*
[3]*School of Mathematics and Computer Science, Ningxia Normal University, Guyuan, China*

Correspondence should be addressed to Jianjun Cheng; chengjianjun@lzu.edu.cn

Received 18 November 2019; Revised 20 February 2020; Accepted 29 February 2020; Published 23 March 2020

Academic Editor: Eric Campos

The centrality plays an important role in many community-detection algorithms, which depend on various kinds of centralities to identify seed vertices of communities first and then expand each of communities based on the seeds to get the resulting community structure. The traditional algorithms always use a single centrality measure to recognize seed vertices from the network, but each centrality measure has both pros and cons when being used in this circumstance; hence seed vertices identified using a single centrality measure might not be the best ones. In this paper, we propose a framework which integrates advantages of various centrality measures to identify the seed vertices from the network based on the TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) multiattribute decision-making technology. We take each of the centrality measures involved as an attribute, rank vertices according to the scores which are calculated for them using TOPSIS, and then take vertices with top ranks as the seeds. To put this framework into practice, we concretize it in this paper by considering four centrality measures as attributes to identify the seed vertices of communities first, then expanding communities by iteratively inserting one unclassified vertex into the community to which its most similar neighbor belongs, and the similarity between them is the largest among all pairs of vertices. After that, we obtain the initial community structure. However, the amount of communities might be much more than they should be, and some communities might be too small to make sense. Therefore, we finally consider a postprocessing procedure to merge some initial communities into larger ones to acquire the resulting community structure. To test the effectiveness of the proposed framework and method, we have performed extensive experiments on both some synthetic networks and some real-world networks; the experimental results show that the proposed method can get better results, and the quality of the detected community structure is much higher than those of competitors.

1. Introduction

Many complex systems can be epitomized as complex networks, in which vertices represent individuals and edges depict the interrelation of them. At present, complex network analysis has been applied in many fields, such as sport competition networks, biological networks, social networks, and political election networks. For these complex networks, community structure is one of their important characteristics. A so-called community is a group of network vertices; edges among vertices inside the group are relatively denser, while edges connecting to the remainder part of the network are relatively sparser [1].

Communities are always corresponding to the functional modules of the real-world systems, such as complexes or pathways in protein-protein interaction networks or metabolic networks, real social groupings with the same occupations, interests, and so forth in social networks. Therefore, we can explore the functional characteristics of the systems via detecting the community structures from the corresponding networks. Moreover, some previous studies [2, 3] have shown that networks own some special characteristics at the community level which differ from those at the individual-vertex level or the level of the entire network. Therefore, some more interesting properties of the network

can be captured through detecting communities. In addition, community detection can help to facilitate many downstream studies, such as prevention of epidemic propagation [4], disease detection [5], link prediction [6], and influence maximization [7]. Overall, the problem of community detection has attracted many researchers from different fields in the last decade.

Many community-detection algorithms [8, 9] have been brought forth; most of them are global ones and suffer from the high time consumptions in most cases, so the local approaches have been the hot spot of research recently due to their efficiency. The seed-expanding methods are a typical kind of local approaches, which firstly identify seeds of communities utilizing various centrality indexes and then expand each of communities by absorbing vertices to join.

Most of the seed-expanding algorithms identify the seed vertices normally using only one single centrality index, but each centrality index has both pros and cons when being used in this circumstance; one centrality that performs well on a kind of networks might embody the poor performance on another kind of networks. Hence, seed vertices identified by using a single centrality index, to some extent, are not the best ones. To take full advantage of every centrality index, we propose a framework in this paper which integrates multiple centrality indexes by using the TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) [10] multi-attribute decision-making technology to identify seed vertices. We take each of the centralities involved as one attribute, rank vertices according to the scores which are calculated for them using TOPSIS, and then take vertices with top ranks as the seeds. To make practice of the framework, we concretize it in this paper by considering four centrality indexes, namely, degree centrality, betweenness centrality, eigenvector centrality, and PageRank centrality, as attributes to select seed vertices from the network. Then, we expand communities by iteratively inserting one unclassified vertex to the community to which its most similar neighbor belongs, and the similarity between them is the largest among all the pairs of unclassified and classified vertices. In this procedure, we need to specify how many seed vertices should be selected; i.e., we need to know the number of communities a priori. However, to determine the exact number of communities contained in a network is still an open question. Thus, we do not invest time to acquire the exact number of communities, but specify its upper bound instead according to our previous work [11], and finally consider a fine-tuning process to merge some small communities to get the resulting community structure. To testify the effectiveness of this method, we have conducted extensive experiments on both some artificial networks and some real-world networks; the experimental results show that the proposed framework is effective, and the concretized method can extract high-quality community structures from networks steadily.

The main contributions of our work are as follows:

(i) We propose a seed vertices identification framework which integrates multiple centrality indexes using TOPSIS; the seed vertices selected by this method take full advantage of every centrality index

(ii) Based on the selected seed vertices, we propose a method to detect communities from networks, which is a seed-expanding method and can detect high-quality community structure without needing to specify the number of communities

(iii) Extensive experiments are carried out to testify the effectiveness and performance of the proposed method

The remainder of this paper is organized as follows. Section 2 reviews some literature about community detection, the details of the proposed framework and concretized method are elaborated in Section 3, Section 4 presents the experimental results and analysis on both synthetic networks and real-world networks, and finally we make a conclusion in Section 5.

2. Related Work

Many community-detection algorithms have been proposed in the last decades. The hierarchical clustering methods are the ones developed in the early period; GN [1, 12] and Fast Q [13] are the representative algorithms of this kind. The former works in a divisive way, which iteratively removes the edge with the largest betweenness from the network, until all the edges are removed; the latter operates in an agglomerative approach, which takes every vertex as a single-member community first and then repeatedly merge two communities until all the vertices are assigned to the same community. WMW [14] defines a new dynamic structural similarity index and applies it to a heuristic agglomerative hierarchical algorithm which not only merges clusters with maximal similarity, but also merges clusters that do not meet the parameterized community definition to extract communities. In addition to this, some algorithms try to integrate the divisive way and agglomerative approach to detect communities. For instance, He et al. [15] proposed a method identifying stepwise communities from temporal networks, which divides the communities of the previous time step into small modules first, then constructs a small network by taking each module as a node, and next extracts communities for the current time step from the newly constructed network. Besides this, the works in [16, 17] are also the algorithms of this type; they all split the network into small vertex groups and then merge some of them into larger ones to get the resulting community structure. In general cases, the outputs of the hierarchical clustering methods are a dendrogram; the hierarchy corresponding to the largest modularity is taken as the final result.

The modularity [12] is an index proposed by Newman and Girvan along with algorithm GN; it is always used to measure the strength of the community structure; the larger value of modularity indicates the higher quality of the result. Its physical meaning leads to another kind of algorithms, which try to detect communities via optimizing the modularity function to seek its maximum value. For example, Fast Q iteratively joins a pair of communities whose merging can lead to the largest modularity increase. Louvain [18] proposes a vertex moving strategy which calculates the

modularity gain of moving a vertex from its own community to the adjacent one, moves every vertex into the community with the largest positive gain, and then takes each community as a super-vertex with self-loop and repeats the vertex moving process until every vertex cannot be moved. CONCLUDE [19] maps vertices into a Euclidean space by computing the centrality for each edge using non-backtracking random walks of finite length, then computes all-pair distances among vertices in the Euclidean space, and adopts those distances as weights of edges, and finally uses Louvain to acquire community structure via maximizing the modularity. Besides this, evolutionary algorithms have also been exploited to detect communities by taking the modularity function as the objective to be optimized [20].

In 2007, Raghavan et al. utilized the information propagation mechanism and proposed the LPA (Label Propagation Algorithm) [21], which assigns a unique label for each vertex in the network initially; then, every vertex updates its own label to be the one which occurs most frequently among its neighbors, with ties being broken uniformly randomly. In this way, vertices can quickly reach a consensus on their labels, and vertices with the same label form a community at last. LPA is very simple and easy to implement and can work with high efficiency, as it has a near linear time complexity. Just because of these advantages, several variations have also been proposed since then. LPAm [22] modifies the label-update rule to maximize the modularity of the detected community structure; its result is more deterministic than that of the basic LPA. Xie and Szymanski [23] enhanced LPA by introducing new label-update rule and label propagation criterion, considering only the vertices whose labels need to be updated, and for each of those vertices, considering the neighborhood strength when determining its new labels. These considerations improve not only the efficiency by avoiding unnecessary updates, but also the qualities of the result community structure. Chin and Ratnavelu [24] used the number of mutual neighbors to get the main communities first and then exploited an improved LPA with some independent constraints to insert the remainder vertices into communities.

The density-based methods migrate the concept of "density" used in clustering analysis to the problem of community detection and use this concept to detect communities from the network. SCAN [25] defines the concepts of "direct structure reachability," "structure reachability," and "structure connectivity" to detect communities, hubs, and outliers, where a community is a structure connected clustering. SCAN++ [26] introduces a new data structure and reduces the number of density evaluations by computing the density for the two-hop-away adjacent vertices only, so that it works with a lower time consumption. Another kind of density-based methods is based on the density peak clustering algorithm, Fdp [27]. For instance, IsoFdp [28] maps vertices in the network as points in a low-dimensional manifold and then gets communities by clustering through Fdp. LCCD [29] exploits Fdp to identify the structural centers from the network and then acquires the results by expanding communities from the center vertices to the borders using a local search approach.

Network dynamics based methods simulate the dynamic procedure on the network to detect communities. Walktrap [30] performs random walk to compute the structural similarity between vertices and between communities and then repeatedly merges a pair of communities under the guidance of the distance which is defined using the similarity to acquire the resulting community structure. RWA [31] calculates the probability of a vertex belonging to each community based on random walks and then expands each of communities by absorbing the vertex which has the largest probability to belong to it. Attractor [32] proposes the concept of distance dynamics to detect communities. The distance influences the interaction between vertices, and the interaction will always change the distance; such interplays render vertices in the same community to converge together and those in different communities diverge each other gradually. BiAttractor [33] extends the distance dynamics to bipartite networks for detecting two-mode communities.

As aforementioned, most of these algorithms are global ones; high time complexity is one of their disadvantages, but local methods can overcome this downside. The seed-expanding methods are a typical kind of local approaches, which identify seeds of communities first utilizing various centrality indexes and then expand the communities by absorbing vertices to join according to some rules. For instance, the aforementioned IsoFdp [28] identifies seed vertices by finding points which are vertices mapped into a low-dimensional manifold with density peaks and then assigns other vertices to the nearest seeds. RWA [31] uses the degree centrality to select seed vertices from the network; it takes local maximum degree vertices as seeds of communities and then expands each of communities by adding the vertex which is most likely to belong to it repeatedly. Shang et al. [34] proposed a community-detection algorithm which uses the degree centrality index to select the core vertices, then expands communities to include neighbor vertices which have larger similarity with the core vertex, and finally integrates some communities based on the proposed modularity density increment. ECES [35] identifies core vertices using a proposed index, core-dominance, which is defined for every vertex as the sum of the extended Jaccard similarity between the vertex and other vertices, and then adds vertices whose membership degree to the community is larger than a given threshold into the community. K-rank-D [36] employs a decision graph about PageRank centrality and minimum distance to select seed vertices, which have larger centrality values and are dispersedly located in the network, and then uses K-Means to cluster other vertices.

3. The Proposed Method

As aforementioned, the proposed framework and method identify the seed vertices from the network by utilizing the TOPSIS multiattribute decision-making technology first, then take each of the selected seeds as a community, expand communities by attracting the unclassified vertex which is most similar to the one being classified into a certain community to join that community iteratively, and then employ a community-merging procedure to agglomerate

some of the initial communities to acquire the final result at last. The pseudocode outlining the framework is listed in Algorithm 1, which implements the above steps loyally.

3.1. Selection of Seed Vertices Based on TOPSIS. In Algorithm 1, the function call TOPSIS() uses the TOPSIS multiattribute decision-making technology to select seed vertices. TOPSIS is one of the typical multiattribute decision-making technologies, which was proposed by Hwang and Yoon in 1981, and has become one of the most popular multiattribute decision-making methods.

It considers a finite set of attributes and takes each attribute as a potential choice to search for the optimal alternative, which is the solution with the largest distance to the negative ideal one and the shortest distance to the positive ideal one. In decision-making process, some attributes are profit ones, which make positive contributions to the solution, whereas some indicate cost, which have negative impacts on the solution. The positive ideal solution achieves the maximum of the profit and the minimum of the cost, while the negative ideal solution maximizes the cost and minimizes the profit on the contrary. In Algorithm 1, we take each of the centralities as an attribute, utilizing the TOPSIS to select the seed vertices by integrating multiple centralities. The steps of the procedure are listed in Algorithm 2.

In this algorithm, steps 1 through 6 are the general procedure of TOPSIS multiattribute decision-making technology, step 7 selects the top-K vertices with the largest score values which is calculated by using the TOPSIS technology, and the selected seed vertices are returned at the last step. All the steps in this algorithm are intuitive and almost self-explanatory, needing no further interpretation.

Every centrality reflects the vertex importance from a certain perspective; this algorithm integrates all the centralities involved to calculate a score for each vertex. This score can make full utilization of the structural characteristics of the vertex and the network; thus, the seed vertices which are selected according to the score are the best ones in principle if the appropriate centralities are considered.

3.2. Expansion of Seed Communities. After the seed vertices are selected using the TOPSIS technology, we first take each of the seed vertices as a separate community and then expand the seed communities by absorbing the unclassified vertices into them iteratively. In Algorithm 1, the function call Expanding() is responding for the community-expansion procedure.

In real-world systems, a community is always corresponding to a group of entities having some common characteristics; hence vertices in the same community tend to be more similar to each other than to vertices located in the other communities. Inspired by this, we expand the communities using the similarity between vertices as a criterion. Specifically, we iteratively select the most similar pair of vertices—one of them has been classified into a community and the other one is still unclassified into any community—and insert the unclassified one into the community to which its most similar buddy belongs, until every vertex has a definite community affiliation. This procedure is implemented as the pseudocode shown in Algorithm 3.

In this algorithm, operations in steps 1 and 2 are the preparations for community expansion, step 3 selects a pair of vertices with the largest similarity between the classified vertices and the unclassified vertices, step 4 inserts the selected unclassified vertex into the community in which its most similar classified partner has been, and step 5 repeats the "select-insert" operations until every vertex has been assigned to a certain community. At that time, the initial community structure is obtained and returned by the last step.

The similarity between vertices plays an important role in the community-expansion procedure. Here, we calculate the similarity between vertices u and v as the following equation:

$$\mathrm{sim}(v, w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\min\{d(v), d(w)\}}, \tag{1}$$

where $\Gamma(v)$ and $\Gamma(w)$ denote the sets of neighbors of vertices v and w, respectively, and $\mathrm{d}(v)$ and $\mathrm{d}(w)$ denote the degree of the two vertices, respectively.

3.3. Merging of the Initial Communities. After calling the function Expanding(), we obtain the initial community structure. However, just as mentioned before, because of the hardness of determining the exact number of communities, we specify the upper bound of the seed number when running Algorithm 2; this leads to the result that Algorithms 2 and 3 detect more communities than expected. In addition, we also find that some communities are too small to make sense in practice. To overcome these problems, we add postprocessing to merge some initial communities to acquire the resulting community structure; the function call merge() in Algorithm 1 accomplishes this task.

We need some criterion to determine which communities should be merged; as the exact number of communities is unknown, the termination of the merge procedure cannot depend on that number. Here, we associate these problems with the metric measuring the quality of the resulting community structure. For measuring the quality of the community structure, the modularity (denoted as Q) [12] is a widely used index, which is defined as follows. For a community structure containing k communities $\mathrm{CS} = \{C_1, C_2, \ldots, C_k\}$, we define a $k \times k$ matrix $\mathbf{e}$, whose element e_{ij} is the fraction of edges between communities C_i and C_j to the total edges in the network. We denote the sum of row i in $\mathbf{e}$ as $a_i = \sum_{j=1}^{k} e_{ij}$, which is the ratio of edges associated with vertices located in community C_i to the total edges in the network, and all the a_i's constitute a k-dimensional vector a. Then, the modularity is defined as

$$Q = \sum_{i=1}^{k} \left(e_{ii} - a_i^2\right). \tag{2}$$

In the proposed framework, we do not directly use the modularity as the community-merge criterion. Instead, we

employ an approach which is similar to the procedure of Fast Q [13] to calculate the increase of the modularity leading to by joining a pair of communities and select the two communities with the largest modularity increase to perform the merge operation. According to the literature [13], the modularity increase ΔQ of merging communities C_i and C_j is

$$\Delta Q = 2\left(e_{ij} - a_i a_j\right), \tag{3}$$

which can be calculated efficiently. Furthermore, we terminate the merging procedure when the largest modularity increase ΔQ is not positive anymore, because joining the two communities with the negative modularity increase will deteriorate the quality of the resulting community structure.

On the basis of the above discussion, the community-merging procedure is shown in Algorithm 4. In this algorithm, steps 1 and 2 are both the preparations for community merging, step 3 calculates the modularity increases for all pairs of communities, step 4 selects the pair of communities with the largest modularity increase, step 5 performs the real merging operation for the two selected communities, step 6 recalculates matrix **e** and vector a to reflect the merging effect and prepares for calculating the modularity increase in the next iteration, and step 7 repeats steps 3 through 6 until it cannot select any pair of the communities with the positive modularity increase. Finally, the resulting community structure is obtained and is returned by the last step.

3.4. Time Complexity. The proposed framework is constituted by four steps of operations; hence, its time consumption is also comprised of costs of the four steps. In Algorithm 1, function call TOPSIS() selects seed vertices using TOPSIS multiattribute decision-making strategy; it is implemented in Algorithm 2, in which all of steps 1, 2, and 4 take $O(nk)$ times to accomplish. Step 3 can be completed in $O(k \log n)$ by organizing the attributes in a max-heap data structure, step 5 takes $O(n)$ times, and steps 6 and 7 can also utilize a max-heap to accomplish the tasks with the cost of $O(n \log n)$ and $O(K \log n)$, respectively. In general, $k \ll K \ll n$, so the time complexity of TOPSIS() is $O(n \log n)$.

The operation in step 2 of Algorithm 1 is intuitive and can be done in $O(K)$. For function call Expanding(), the operations are listed in Algorithm 3. This is an iterative procedure; in each iteration, its time consumption is concentrated in step 3; it takes approximately $O(\log n)$ times to select the most similar pair of vertices. Each iteration selects one unclassified node and inserts it into the corresponding community; therefore, the total time complexity of the community-expansion procedure is $O(n \log n)$.

As to function call merge(), Algorithm 4 describes its logic. It begins with K initial communities; then the initialization cost of step 1 is $O(K)$. The matrix **e** and vector a can be calculated by traversing edges of the network; step 2 takes $O(m)$ times. Step 3 calculates modularity increases; only adjacent communities need to be considered, they can be calculated via visiting intercommunity edges in $O(m/K)$ in average, and step 4 selects the pair of communities to be merged with cost of $O(\log(m/K))$ consequently. The merging operation in step 5 can be executed in $O(1)$, and the update of **e** and a can be performed with $O(K)$ time consumption. In the worst case, steps 3–6 need to be repeated K times; therefore, the time complexity of merge procedure is $O(m) + O(m/K \cdot K) + O(K) + O(K^2) \sim O(m)$.

In summary, the total time consumption of the proposed framework is $O(n \log n) + O(n \log n) + O(m) \sim O(m)$.

4. Experiments and Results

4.1. Networks and Comparison Systems. To testify the effectiveness and the performance of the proposed framework, we have conducted extensive experiments on both some of artificial networks and some of real-world networks. The artificial networks are generated using the LFR benchmark network creator [37], which has some options to control the characteristics of the generated networks. In our experiments, we adjust those options to produce three series of networks; the details of the options are listed in Table 1, in which μ is a key option to adjust the ratio of edges which are associated with every vertex connecting outside of its community to the total edges associated with that vertex; the smaller values of μ will produce networks with well-separated communities, while the larger μ will lead to fuzzier boundaries among communities. In this section, we vary the values of μ from 0.1 to 0.9 by increasing 0.1 each time to generate the three series of networks.

In addition to the experiments on artificial networks, we have also carried out extensive experiments on 11 real-world networks, whose statistical information is listed in Table 2.

Up to now, our proposal is a framework rather than a specific community-detection method, because centralities which should be employed in Algorithm 1 for performing the TOPSIS procedure are not specified. To carry out the experiments, we need to concretize the framework to be a community-detection method. In theory, the combination of any number of centralities can be integrated into the framework. Here, we consider four widely used indexes, namely, the degree centrality, the betweenness centrality (for vertex), the eigenvector centrality, and the PageRank centrality to carry out experiments; **C** in Algorithms 1 and 2 contains the four centralities, and the value of k is 4 naturally. Clearly, all the four centralities are profit attributes for the identification of seed vertices. For the other two parameters, K is the upper bound of the community number; we relax it to be $\sqrt{n}$, the same as the strategy we used in our previous work [11], and w to be an all 1 vector; i.e., $w = \{1, 1, \ldots, 1\}$.

Besides this, we have also compared the results which are detected by our proposed method with those which are extracted by 6 state-of-the-art community-detection algorithms, namely, Fast Q [12], LPA [21], Walktrap [30], Attractor [32], IsoFdp [28], and WMW [14]; all of them have been introduced in Section 2. The comparison results will be presented and analyzed in Section 4.3.

4.2. Evaluation Metrics. In our experiments, we use two indexes to measure the quality of the detected community

Input: $G(V, E)$, the network; K, the upper bound of the community number; $\mathbf{C}$, k centrality functions; w, k−dimensional vector of weight for centralities.
Output: CS: the detected community structure.
(1) Select the K seed vertices using TOPSIS multiattribute decision-making technology: $S \longleftarrow$ TOPSIS (G, C, K, w)
(2) Take each vertex $v \in S$ as the first seed of a community: $\mathrm{CS}_{\mathrm{seed}} \longleftarrow \{\{v\} \mid v \in S\}$;
(3) Expand communities in the community structure $\mathrm{CS}_{\mathrm{seed}}$: $\mathrm{CS}_{\mathrm{init}} \longleftarrow$ Expanding $(G, \mathrm{CS}_{\mathrm{seed}})$
(4) Merge some of the initial communities in $\mathrm{CS}_{\mathrm{init}}$ to acquire the resulting community structure: $CS \longleftarrow$ merge $(G, \mathrm{CS}_{\mathrm{init}})$
(5) return CS

Algorithm 1: The steps of the proposed framework.

Input: $G(V, E)$, the network; K, the upper bound of the community number; $\mathbf{C}$, k centrality functions; w, k−dim weight vector for centralities.
Output: S, the set of seed vertices.
(1) Organize all the vertices with the k centrality functions $c_1, c_2, \ldots, c_k$ $(c_j \in \mathbf{C}, j = 1, 2, \ldots, k)$ as an attribute matrix $\mathbf{P}$: $\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1k} \\ p_{21} & p_{22} & \cdots & p_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nk} \end{bmatrix}$.
where $n = |V|$ is the vertex number, $p_{ij} = c_j(i)$ $(i = 1, 2, \ldots, n; j = 1, 2, \ldots, k)$.
(2) Normalize each column of matrix $\mathbf{P}$, and weight it as $a_{ij} = (p_{ij} / \sqrt{\sum_{r=1}^{n} p_{rj}^2}) \cdot w_j$ $(i = 1, 2, \ldots, n; j = 1, 2, \ldots, k)$, to yield a new attribute matrix $\mathbf{A}$: $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nk} \end{bmatrix}$.
(3) Calculate the positive and the negative ideal solutions s_j^p and s_j^n for attribute j: $s_j^p = \begin{cases} \max\{a_{ij} \mid i = 1, 2, \ldots, n\}, & j \in A_p, \\ \min\{a_{ij} \mid i = 1, 2, \ldots, n\}, & j \in A_c, \end{cases}$
$s_j^n = \begin{cases} \min\{a_{ij} \mid i = 1, 2, \ldots, n\}, & j \in A_p, \\ \max\{a_{ij} \mid i = 1, 2, \ldots, n\}, & j \in A_c. \end{cases}$
where A_p and A_c represent the sets of profit attributes and cost attributes, respectively.
(4) Calculate the distances for each vertex i to its positive ideal solution and negative ideal solution: $\mathrm{dist}_i^p = \sqrt{\sum_{j=1}^{k} (a_{ij} - s_j^p)^2}$, $\mathrm{dist}_i^n = \sqrt{\sum_{j=1}^{k} (a_{ij} - s_j^n)^2}$.
(5) Calculate a score for each vertex indicating the relative closeness to the ideal solution: $\mathrm{score}_i = \mathrm{dist}_i^n / (\mathrm{dist}_i^n + \mathrm{dist}_i^p)$.
(6) Rank all the vertices in descending order by the score calculated in step 5.
(7) Select K vertices with the top-K scores:$S \longleftarrow$ Top_$K(V$, score, $K)$
(8) return S

Algorithm 2: TOPSIS: seed vertices selection using the TOPSIS multiattribute decision-making technology.

structures: one is the modularity Q [12] and the other is the NMI (Normalized Mutual Information) [46]; both of them are widely used to evaluate the performance of the community-detection algorithms.

The modularity has been defined as equation (2); we reformulate it here as

$$Q = \sum_{i=1}^{k} e_{ii} - \sum_{i=1}^{k} a_i^2, \tag{4}$$

where the first term $\sum_{i=1}^{k} e_{ii}$ is the ratio of edges inside of communities and the second term $\sum_{i=1}^{k} a_i^2$ is the expected value for the same ratio, which is calculated on a null model graph. Such graph is constructed by taking the same vertices of the original network and keeping the same degree distributions for every vertex, but edges between vertices are connected randomly. That is to say, the modularity measures the quality of the detected community structure from the perspective that how far it deviates from a random graph, the further it departs from the random graph, the stronger the community structure is, the larger Q is to be, and the effective value of Q is in the range of [0, 1].

The NMI is defined for two partitions $CS = \{C_1, C_2, \ldots, C_k\}$ and $P = \{P_1, P_2, \ldots, P_{k'}\}$; it is calculated as follows:

$$\mathrm{NMI} = \frac{-2 \sum_{i=1}^{k} \sum_{j=1}^{k'} n_{ij} \log\left(n_{ij} \cdot n / n_i^C \cdot n_j^P\right)}{\sum_{i=1}^{k} n_i^C \log\left(n_i^C / n\right) + \sum_{j=1}^{k'} n_j^P \log\left(n_j^P / n\right)}, \tag{5}$$

where $n_{ij} = |C_i \cap P_j|$, $n_i^C = |C_i|$, and $n_j^P = |P_j|$, separately.

In practical usage, partitions P and CS are always corresponding to the ground-truth community structure and the detected one, so the NMI measures the ability of a community-detection method from the perspective that how much its detected community structure is close to the ground truth. The value of NMI also ranges in [0, 1], larger being better.

Input: $G(V, E)$, the network; CS_{seed}, the seed communities
Output: CS_{init}, the initial community structure
(1) Initialize the initial community structure CS_{init} to be CS_{seed}: $CS_{init} \longleftarrow CS_{seed}$
(2) Collect the vertices remain unclassified: $U \longleftarrow V - \{v \mid v \in C_i, C_i \in CS_{init}, i = 1, 2, \ldots, |CS_{init}|\}$
(3) Select the most similar pair of vertices, one of them has been assigned to a certain community, the other one is still unclassified: u, $i \longleftarrow \arg \cdot \max_{v,j}\{\mathrm{sim}(v, w) \mid v \in U,\ w \in C_j,\ C_j \in CS_{init}\}$
(4) Insert the selected unclassified vertex into the corresponding community: $C_i \leftarrow C_i \cup \{u\}$; $U \longleftarrow U - \{u\}$
(5) Repeat steps 3 and 4 until all the vertices are inserted into some communities, at that time, $U = \phi$
(6) **return** CS_{init}

Algorithm 3: Expanding: community expansion by absorbing the unclassified vertex to join the community to which the most similar buddy of that vertex belongs.

Input: $G(V, E)$, the network; CS_{init}, the initial community structure
Output: CS, the resulting community structure
(1) Initialize the community structure CS to be CS_{init}:
$CS \longleftarrow CS_{init}$
(2) Calculate matrix **e** and vector a for community structure CS.
(3) Calculate the modularity increase ΔQ_{ij} for any pair of communities C_i and C_j:
for $C_i \in CS$, $C_j \in CS(i < j)$ **do**
$\Delta Q_{ij} \longleftarrow 2(e_{ij} - a_i a_j)$
end
(4) Select the pair of communities with the largest modularity increase:
$r, s \longleftarrow \arg \max_{i,j}\{\Delta Q_{ij}\}$
(5) Merge the two selected communities if the modularity increase is positive:
if $\Delta Q_{rs} > 0$ **then**
$k \longleftarrow |CS|$; $C_{k+1} \longleftarrow C_r \cup C_s$;
$CS \longleftarrow CS \cup \{C_{k+1}\} - \{C_r, C_s\}$
end
(6) Update matrix **e** and vector a to reflect the effect of the merge.
(7) Repeat steps 3–6, until step 4 cannot select any pair of communities whose merge leads to the positive modularity increase.
(8) return CS

Algorithm 4: Merge: merging of some of the initial communities.

4.3. Experiments on Artificial Networks. We have carried out experiments on three series of artificial networks to test the effectiveness of the proposed method. As aforementioned, the artificial networks are produced using the LFR benchmark network generator; the options which are used to generate the networks are listed in Table 1. Among them, the mixing parameter μ is a critical one, which adjusts the fraction of edges associated with every vertex connecting outside of its community. In this group of experiments, we change the values of μ from 0.1 to 0.9 by increasing 0.1 each time, generate 10 networks for each value of μ, and keep the same settings for other options of LFR network generator. We run our method and the comparison algorithms on each of them and take the average of NMI as the final metric to evaluate the performance of the proposed method and the comparison algorithms. The comparison results are shown in Figures 1(a)–1(c), respectively.

In Figure 1(a), the proposed method detects the exact community structures which perfectly match with the ground truth from networks with $\mu \leq 0.4$; hence, its NMI values are maximal, 1, which are the same as those of Walktrap and IsoFdp. In the range of $0.5 \leq \mu \leq 0.6$, although its NMI values are smaller than those of Walktrap and IsoFdp, they are still larger than those of Fast Q, LPA, and Attractor, and the values are rather large. Regarding comparison algorithms, Fast Q performs the worst during $\mu < 0.5$; its NMI values are the smallest on the networks. When $\mu \geq 0.5$, its performance is only better than that of LPA. LPA also extracts the exact community structures from networks with $\mu \leq 0.3$, but the quality of its detected results drops dramatically when $\mu > 0.4$; the reason might be that vertices in the networks have more edges connecting outside of its communities along with the increase of μ. Thus, LPA tends to update the labels of vertices to be the incorrect ones, leading to low-quality results. For Attractor, it reveals community structures which are identical to the ground truth as well from networks with $\mu \leq 0.3$ and performs the best when $\mu \geq 0.7$, but its NMIs are smaller than those of our method in the range of $0.3 < \mu \leq 0.6$, indicating its performance is not stable. For algorithm WMW, its performance is slightly higher than those of our method when $\mu = 0.6$ only; in most cases, our method is better. Walktrap and IsoFdp perform the best on the networks when $\mu \leq 0.6$; their values are the largest among all the algorithms.

Table 1: The options of LFR network generator used for producing the 2 series of artificial networks. n is the number of vertices; d_{max} and $\langle d \rangle$ are the maximum and the average degree of vertices; exp_d and exp_{com} are the exponents of the power-law distributions followed by the degree and the community size; C_{min} and C_{max} are the minimum and maximum number of vertices in a community; d_G is the density of the network; μ is the mixed parameter, which controls the ratio of edges for each vertex connecting outside of the community to the total edges associated with that vertex.

Network	n	d_{max}	$\langle d \rangle$	exp_d	exp_{com}	C_{min}	C_{max}	d_G	μ
LFR_1000	1000	50	20	−2	−1	20	100	0.03800–0.04000	0.1–0.9
LFR_5000	5000	50	20	−2	−1	20	100	0.00770–0.00790	0.1–0.9
LFR_10000	10000	50	20	−2	−1	20	100	0.00386–0.00388	0.1–0.9

Table 2: The details of the 11 real-world networks used in our experiments.

Network	Vertices	Edges	# of communities	Reference
Karate club	34	78	2	[1]
Dolphin social network	62	159	4	[38]
Scientists collaboration	118	197	6	[1]
Football game schedule	115	613	12	[1]
Les Mis.	77	253	—	[39]
Jazz	198	2742	—	[40]
E. Coli	423	519	—	[41]
E-mail	1133	5451	—	[42]
PolBlogs	1490	19090	—	[43]
Yeast	2361	7182	—	[44]
PGP	10680	24316	—	[45]

In Figure 1(b), the performance of Fast$Q$ and Walktrap is almost the same as those in Figure 1(a); the former performs the worst and the latter performs the best on networks during $\mu \leq 0.6$; the quality of Fast$Q$ is only better than that of LPA and the NMI values of Walktrap are smaller than those of Attractor on networks during $\mu \geq 0.7$. The trend of the curve for Attractor is also almost the same as that in Figure 1(a), except for the fact that the value of NMI is larger than that of our method when $\mu = 0.4$ in Figure 1(b). For LPA, it performs best on networks during $\mu < 0.6$, which is identical to Walktrap, and even better than Walktrap when $\mu = 0.6$. But it also detects the poor results from networks during $\mu \geq 0.7$ for the same reason as those in Figure 1(a). Different from Figure 1(a), WMW's performance is not better than that of the proposed method on all of the networks. On this series of networks, although the results detected by the proposed method are not the best, they are rather large while $\mu \leq 0.6$, which indicates that the detected community structure is still acceptable.

In Figure 1(c), most of the algorithms perform similarly to those in Figure 1(b), except for Attractor. On this series of networks, Attractor gets the worst results when $\mu < 0.5$; this is contrary to that in Figure 1(b). For our method, its performance is also like that in Figure 1(b); the NMI values are not the best on all the networks, but its values are rather high, even larger than 0.9 when $\mu < 0.6$. They are still larger than those of LPA, Fast$Q$, and IsoFdp while $\mu > 0.6$. These results once again suggest that all the results discovered by our method are acceptable.

Besides this, comparing curves of our method in Figures 1(a)–1(c) shows that the scale of the network has little influence on the quality of the detected results, and the proposed method can detect the high-quality community structure steadily.

4.4. Experiments on Real-World Networks. We have also conducted experiments on 11 real-world networks, which have been listed in Table 2. These networks can be categorized as two groups: the first group involves the first 4 networks, whose ground-truth community structures have been known in advance; the second one includes the remaining 7 networks, which have no publicly accepted ground-truth community structures.

4.4.1. Networks with Ground-Truth Community Structure. We have known the ground-truth community structure of this group of networks; therefore, we evaluate the performance of the proposed method and comparison algorithms using both the modularity and the NMI. The values of these two indexes detected by the algorithms are recorded in Table 3.

On these networks, the proposed method detects either the largest modularity Q or NMI, or both of them, suggesting that the proposed method can detect the high-quality community structure from these networks; its performance on the karate club network and the football game schedule network is not the best only in terms of NMI. For those two networks, only IsoFdp and Attractor obtain the largest NMI on the karate club network and the football game schedule network once, respectively. Fast Q, Walktrap, LPA, and WMW cannot manage to acquire the results with the largest measures any time. This comparison already reflects the superiority of the proposed method to some extent; it can detect the high-quality community structures from networks and performs better than comparison algorithms.

In addition, as the scale of this group of networks is not too large, we can visualize the results intuitively. Below, we

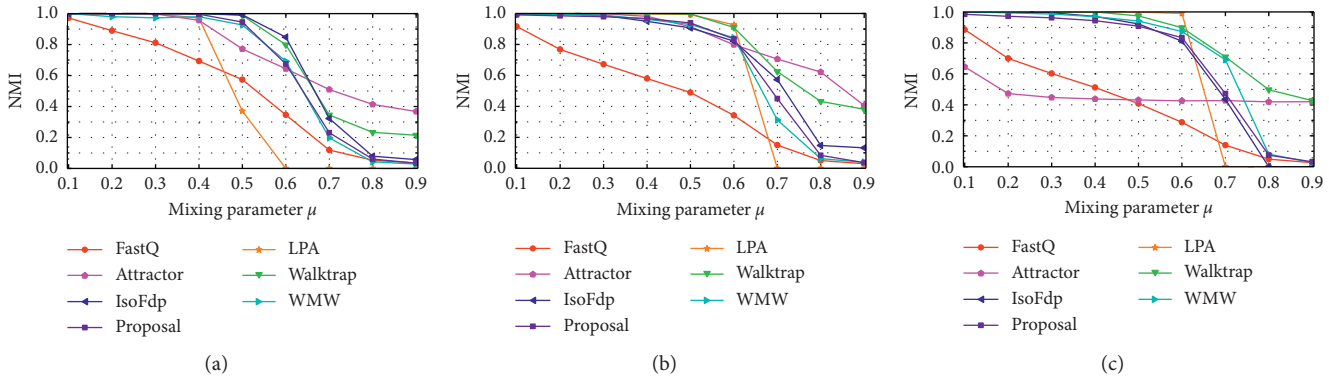


Figure 1: Comparison of results detected by the proposed method and comparison algorithms. (a) The results detected from the networks containing 1000 vertices. (b) The results extracted from the networks containing 5000 vertices. (c) The results discovered from the networks containing 10000 vertices.

Table 3: The modularity and the NMI detected from networks with ground-truth community structure by the comparison algorithms and the proposed method.

Network	Metric	Ground-truth	FastQ	Walktrap	IsoFdp	LPA	Attractor	WMW	Proposal
Karate	Q	0.371	0.381	0.353	0.371	0.385	0.405	0.398	**0.417**
	NMI	—	0.693	0.504	**1.00**	0.622	0.640	0.538	0.696
Dolphin	Q	0.519	0.492	0.489	0.505	0.464	0.495	0.503	**0.528**
	NMI	—	0.719	0.632	0.744	0.710	0.691	0.802	**0.930**
Football	Q	0.601	0.550	0.603	0.599	0.589	0.601	0.533	**0.605**
	NMI	—	0.751	0.954	0.982	0.945	**0.989**	0.954	0.966
Collaboration	Q	0.739	0.749	0.733	0.668	0.638	0.707	0.668	**0.751**
	NMI	—	0.867	0.818	0.825	0.741	0.857	0.743	**0.877**

The largest values on each network are in bold.

present and analyze the detected community structure on each of these networks individually.

(1) The Karate Club Network. This is a most commonly used network in the study of community detection. The 34 vertices represent 34 members of a karate club, and 78 edges represent relationships among these members. Due to a dispute that has arisen between the administrator and the instructor, the club eventually split into two fractions, forming the ground-truth community structure as shown in Figure 2(a).

The community structure detected by the proposed method from this network is shown in Figure 2(b). Compared with the ground-truth community structure, the proposed method detects 4 communities from the network. Although it deviates from the ground truth in some way, it corresponds to a larger modularity than the ground truth. In other words, the quality of the detected result is higher than that of the ground truth.

(2) The Dolphin Social Network. This network contains 62 vertices and 159 edges; vertices represent dolphins living in Doubtful Sound, New Zealand, and edges represent the co-occurrences of pairs of dolphins being observed more often. The ground-truth community structure of this network contains four communities, which is shown in Figure 3(a).

Figure 3(b) presents the community structure detected by the proposed method from this network, which contains 5 communities. It also differs from the ground truth; vertices "ccl," "zap," "double," "sn89," and "sn100" are assigned in an additional community in Figure 3(b), in which the first three vertices form a clique; they connect more tightly to each other than to the remainder of the network; therefore, they are separated from the network as an independent community; for the last two vertices, they are located at the boundary between two communities in Figure 3(a) originally, and among three communities now. Hence, they are easy to be misclassified; our method allocates them into the newly formed three-vertex community. Measuring from the perspective of evaluation metrics, the values of both the modularity Q and the NMI corresponding to the detected result are the largest among all the comparison algorithms, which shows the superiority of the proposed method over comparison algorithms to some extent.

(3) The Football Game Schedule Network. This network is a schedule of American college football matches held in 2000. The 115 vertices represent 115 football teams participating in the games in 2000 season, and the 613 edges represent the total of 613 games played between those teams. All the teams were arranged in 12 conferences; therefore, there are 12 communities contained in the ground-truth community

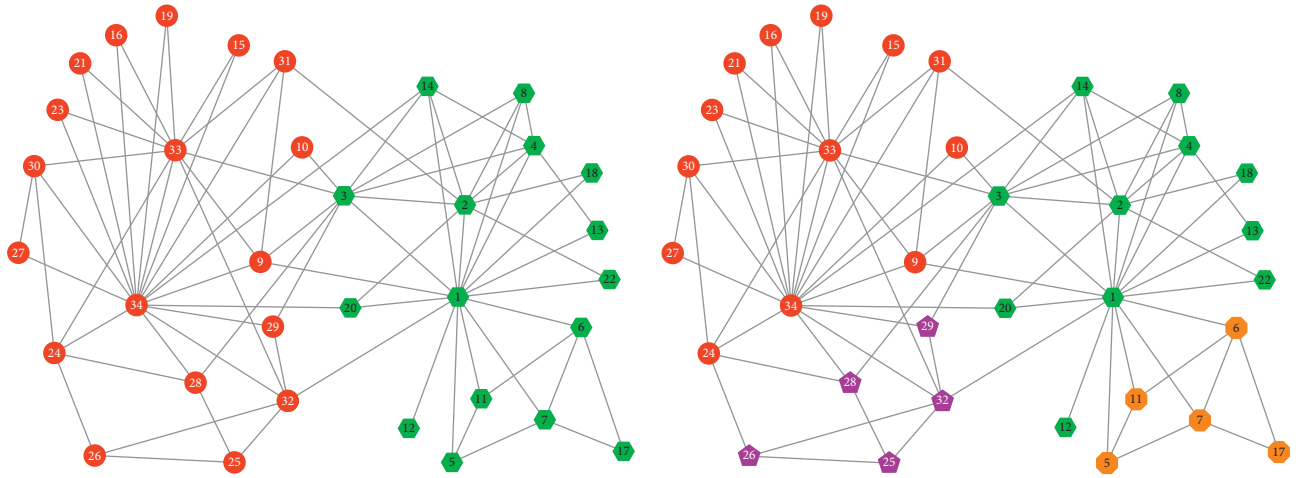


FIGURE 2: The karate club network. (a) The ground-truth community structure. (b) The community structure detected by the proposed method.

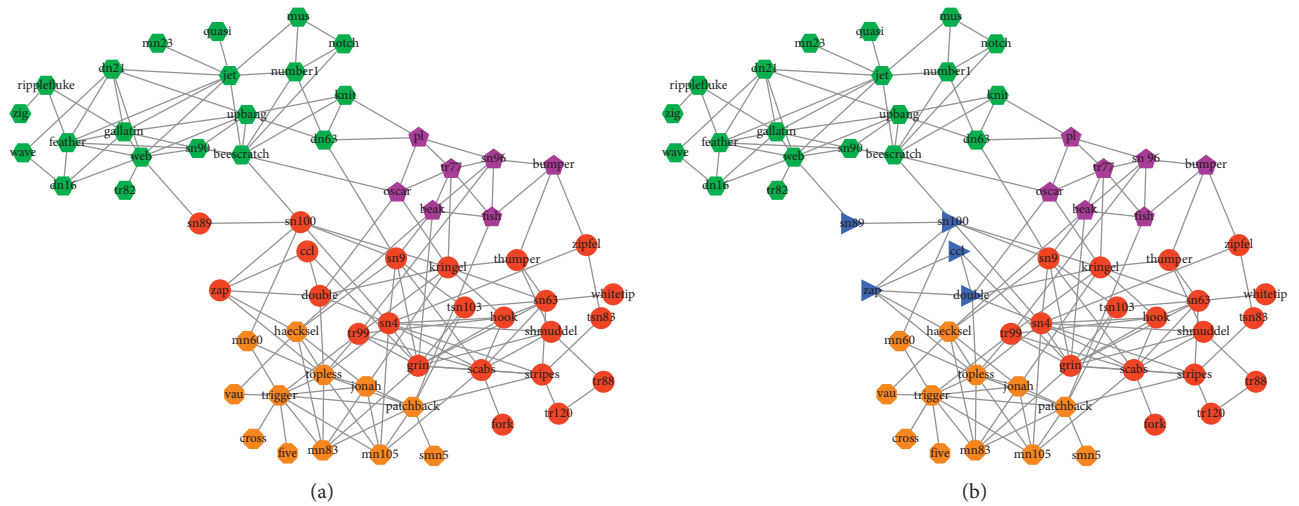


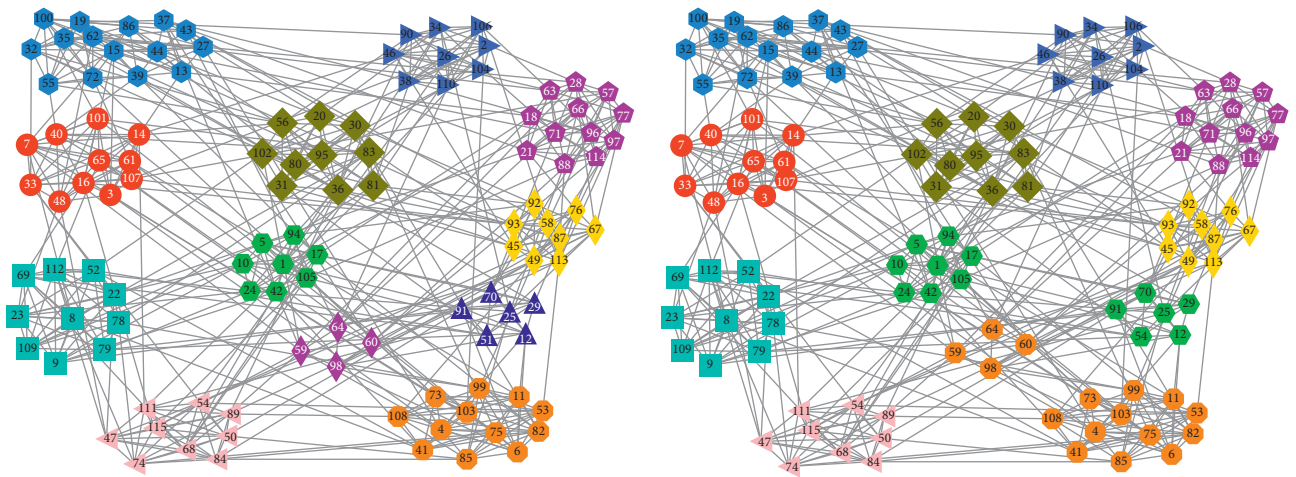
FIGURE 3: The dolphin social network. (a) The ground-truth community structure. (b) The community structure detected by our proposed method.

structure of this network, which is as illustrated in Figure 4(a).

The proposed method detects communities from this network also with a high degree of success; it divides the network into 10 communities, which is exhibited in Figure 4(b), in which two communities involving vertices "60" and "70" are combined into the larger ones containing vertices "6" and "10," respectively. This might be because the two communities are relatively small; teams inside them tend to play more games with teams located in those two communities to which vertices "6" and "10" belong, respectively. Therefore, they are combined into the two larger ones, consequently. Except for this, the other 10 communities detected by the proposed method perfectly match to those in the ground truth. In terms of modularity, the detected result of our method is the largest among those of all comparison algorithms, reflecting the good performance of our method.

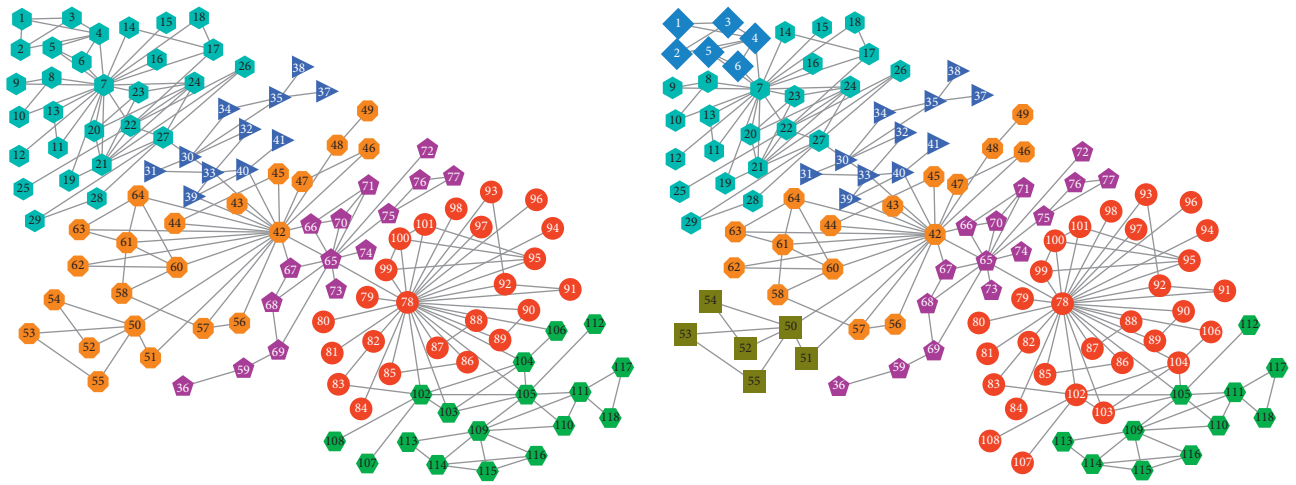
(4) The Scientists Collaboration Network. This network describes the coauthor relationships of some scientific articles, in which 118 vertices represent 118 scientists working at the Santa Fe Institute and 197 edges indicate that 197 pairs of scientists have collaborated on one article at least. According to the different specialities of scientists, this network is naturally divided into 6 communities, which are displayed in Figure 5(a). Feeding this network into the proposed method, we obtain the community structure as shown in Figure 5(b).

Compared to the ground-truth community structure, the proposed method detects 8 communities from this network rather than 6 in Figure 5(a); it isolates two additional communities from the network. These two communities correspond to two relatively small subgraphs; both of them are almost formed by 3 or 4 cliques; i.e., vertices inside them connect to each other more tightly, and their inner edges are much denser than outer ones. Therefore, detecting them as independent communities is



(a) (b)

Figure 4: The network of college football game schedule. (a) The ground-truth community structure. (b) The community structure detected by our proposed method.



(a) (b)

Figure 5: The collaboration network of scientists at the Santa Fe Institute. (a) The ground-truth community structure. (b) The community structure detected by our proposed method.

reasonable. Besides this, 6 vertices (vertices "102," "103," "104," "106," "107," and "108") are classified into the incorrect communities in Figure 5(b); this is because these vertices are also positioned at the boundary between two communities; vertex "78" in the opposite community has larger influence than vertex "105" in their original community. Consequently, vertices "102," "103," and "104" tend to be attracted by "78" to join its community; then, the misclassification of vertices "106," "107," and "108" cannot be avoided. Certainly, this result corresponds to both the largest modularity and NMI among all the comparison algorithms, indicating that the detected result on this network has the highest quality and approaches the ground-truth mostly.

4.4.2. Networks without Ground-Truth Community Structure. This group of networks has no publicly accepted ground-truth communities; therefore, we only use the modularity as the measure metric to evaluate the performance of the proposed method. We take each of them as the input to each comparison algorithm and the proposed method to acquire the community structures from them and then calculate the corresponding modularity using equation (4). The obtained values are filed in Table 4 and visualized intuitively in a bar chart which is illustrated in Figure 6.

On this group of networks, the proposed method obtains the largest modularity on 6 of the 7 networks, suggesting that the detected results on the 6 networks are the best among the comparison algorithms. Only on another network, the quality of the detected result is not

Table 4: The modularity detected from networks without ground-truth community structures by comparison algorithms and the proposed method.

Network	FastQ	Walktrap	IsoFdp	LPA	Attractor	WMW	Proposal
Les Mis.	0.499	0.519	0.510	0.510	0.481	0.532	**0.553**
Jazz	0.439	0.438	0.435	0.377	0.276	0.416	**0.442**
E. Coli	**0.778**	0.746	—	0.691	0.727	0.685	0.774
E-mail	0.510	0.531	0.531	0.398	0.480	0.423	**0.568**
PolBlogs	0.427	0.426	—	0.413	0.323	0.423	**0.431**
YeastL	0.573	0.529	—	0.376	0.517	0.484	**0.620**
PGP	0.854	0.789	0.726	0.765	0.770	0.710	**0.874**

The largest values on each network are in bold.

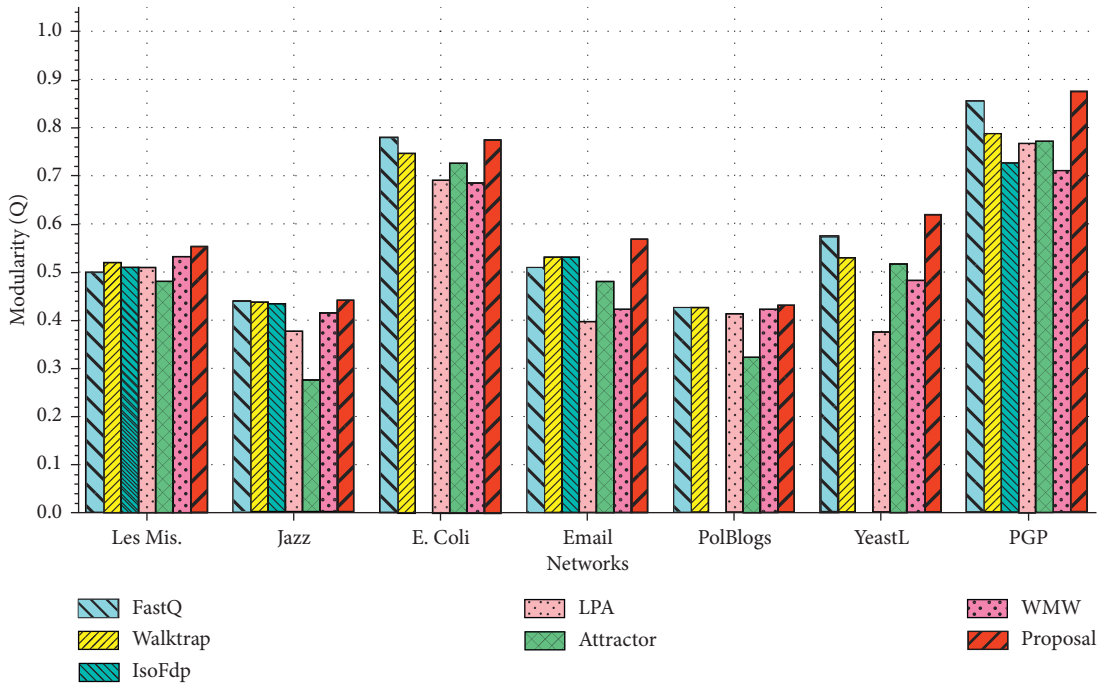


Figure 6: The bar chart of the modularity detected by comparison algorithms and the proposed method from networks without publicly accepted ground-truth community structure.

the highest, but it is still the second best. For comparison algorithms, only Fast*Q* obtains the largest modularity on network E. Coli once; the detected modularity of Walktrap, IsoFdp, LPA, Attractor, and WMW on any network is not the largest. Furthermore, their values of the modularity are smaller than those of our method with a larger difference on most of networks. IsoFdp even cannot manage to extract the effective results from networks E. Coli, PolBlogs, and YeastL, because these networks are not connected, whereas IsoFdp can only extract communities from the connected networks. This result manifests the effectiveness of the proposed framework; the proposed method can detect the high-quality community structures steadily from networks and outperform the competitors significantly.

5. Conclusion

In the seed-expanding based community-detection methods, the selection of seed vertices is vital to the quality of the resulting community structure. Many algorithms exploit the centrality to select seed vertices from the network, but each centrality describes the importance of each vertex from different perspectives; it has both upsides and downsides simultaneously when it is used in such a scenario. Therefore, seed vertices identified using one single centrality are always not the best ones. We attempt to integrate multiple centralities' utilization via the TOPSIS multi-attribute decision-making technology in this paper, and propose a framework to consider multiple centralities by treating each of them as an attribute, ranking vertices according to the scores which are calculated by using TOPSIS, and taking top-rank vertices as the seeds. Then, we also develop a strategy to expand communities based on the selected seeds by attracting the most similar neighbors of vertices inside communities to join.

Next, we get a concretized community-detection method from the proposed framework by considering four concrete centralities. We have performed the experiments about this method on both some synthetic networks and some real-

world networks; the effectiveness of the proposed framework is tested by the experimental results, which also manifests that the concretized method can detect high-quality community structure from networks steadily.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the project of Key R&D Program of Ningxia Province, China (Grant no.: 2018BEE03026).

References

[1] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[2] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E*, vol. 74, no. 3, 2006.

[3] Y. Pan, D.-H. Li, J.-G. Liu, and J.-Z. Liang, "Detecting community structure in complex networks via node similarity," *Physica A: Statistical Mechanics and Its Applications*, vol. 389, no. 14, pp. 2849–2857, 2010.

[4] X. Deng, Y. Wen, and Y. Chen, "Highly efficient epidemic spreading model based lpa threshold community detection method," *Neurocomputing*, vol. 210, pp. 3–12, 2016.

[5] L. Cantini, E. Medico, S. Fortunato, and M. Caselle, "Detection of gene communities in multi-networks reveals cancer drivers," *Scientific Reports*, vol. 5, no. 1, 2015.

[6] Z. Wang, Y. Wu, Q. Li, F. Jin, and W. Xiong, "Link prediction based on hyperbolic mapping with community structure for complex networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 450, pp. 609–623, 2016.

[7] M. Jalayer, M. Azheian, and M. A. M. A. Kermani, "A hybrid algorithm based on community detection and multi attribute decision making for influence maximization," *Computers & Industrial Engineering*, vol. 120, pp. 234–250, 2018.

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

[9] S. Fortunato and D. Hric, "Community detection in networks: a user guide," *Physics Reports*, vol. 659, pp. 1–44, 2016.

[10] C.-L. Hwang and K. Yoon, "Methods for multiple attribute decision making," in *Multiple Attribute Decision Making*, pp. 58–191, Springer, Berlin, Germany, 1981.

[11] J. Cheng, S. Zhao, H. Yang, J. Zhang, X. Su, and X. Chen, "Detecting communities from networks using an improved self-organizing map," *International Journal of Modern Physics C*, vol. 30, no. 6, Article ID 1950054, 2019.

[12] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, 2004.

[13] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, no. 6, Article ID 066133, 2004.

[14] E. Castrillo, E. León, and J. Gómez, "Fast heuristic algorithm for multi-scale hierarchical community detection," in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, ASONAM '17*, Association for Computing Machinery, New York, NY, USA, pp. 982–989, 2017.

[15] J. He, D. Chen, C. Sun, Y. Fu, and W. Li, "Efficient stepwise detection of communities in temporal networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 469, pp. 438–446, 2017.

[16] F. D. Zarandi and M. K. Rafsanjani, "Community detection in complex networks using structural similarity," *Physica A: Statistical Mechanics and Its Applications*, vol. 503, pp. 882–891, 2018.

[17] J. Cheng, L. Li, H. Yang, Li Qi, and X. Chen, "A hybrid spectral method for network community detection," in *Web and Big Data*, Y. Cai, Y. Ishikawa, and J. Xu, Eds., pp. 90–104, Springer International Publishing, Cham, Switzerland, 2018.

[18] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, Article ID P10008, 2008.

[19] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Mixing local and global information for community detection in large networks," *Journal of Computer and System Sciences*, vol. 80, no. 1, pp. 72–87, 2014.

[20] C. Pizzuti, "Evolutionary computation for community detection in networks: a review," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 3, pp. 464–483, 2018.

[21] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, Article ID 036106, 2007.

[22] M. J. Barber and J. W. Clark, "Detecting network communities by propagating labels under constraints," *Physical Review E*, vol. 80, no. 2, 2009.

[23] J. Xie and B. K. Szymanski, "Community detection using a neighborhood strength driven label propagation algorithm," in *Proceedings of the 2011 IEEE Network Science Workshop*, IEEE, New York, NY, USA, pp. 188–195, 2011.

[24] J. H. Chin and K. Ratnavelu, "A semi-synchronous label propagation algorithm with constraints for community detection in complex networks," *Scientific Reports*, vol. 7, no. 1, 2017.

[25] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "Scan: a structural clustering algorithm for networks," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, San Jose, CA. USA, pp. 824–833, August 2007.

[26] H. Shiokawa, Y. Fujiwara, and M. Onizuka, "Scan++," *Proceedings of the VLDB Endowment*, vol. 8, no. 11, pp. 1178–1189, 2015.

[27] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[28] T. You, H.-M. Cheng, Y.-Z. Ning, B.-C. Shia, and Z.-Y. Zhang, "Community detection in complex networks using density-based clustering algorithm and manifold learning," *Physica A: Statistical Mechanics and Its Applications*, vol. 464, pp. 221–230, 2016.

[29] X. Wang, G. Liu, J. Li, and J. P. Nees, "Locating structural centers: a density-based clustering method for community detection," *PLoS One*, vol. 12, no. 1, Article ID e0169355, 2017.

[30] P. Pons and M. Latapy, "Computing communities in large networks using random walks" P. Yolum and T. Güngör Eds.,

in Computer and Information Sciences–IS 2005, pp. 284–293, Springer, Berlin, Germany, 2005.

[31] Y. Su, B. Wang, and X. Zhang, "A seed-expanding method based on random walks for community detection in networks with ambiguous community structures," *Scientific Reports*, vol. 7, no. 1, 2017.

[32] J. Shao, Z. Han, Q. Yang, and T. Zhou, "Community detection based on distance dynamics," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15*, ACM, New York, NY, USA, pp. 1075–1084, 2015.

[33] H.-l. Sun, E. Ch'ng, X. Yong, J. M. Garibaldi, S. See, and D.-b. Chen, "A fast community detection method in bipartite networks by distance dynamics," *Physica A: Statistical Mechanics and Its Applications*, vol. 496, pp. 108–120, 2018.

[34] R. Shang, W. Zhang, L. Jiao, R. Stolkin, and Y. Xue, "A community integration strategy based on an improved modularity density increment for large-scale networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 469, pp. 471–485, 2017.

[35] K. Berahmand, A. Bouyer, and M. Vasighi, "Community detection in complex networks by detecting and expanding core nodes through extended local similarity of nodes," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 4, pp. 1021–1033, 2018.

[36] Y. Li, C. Jia, and J. Yu, "A parameter-free community detection method based on centrality and dispersion of nodes in complex networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 438, pp. 321–334, 2015.

[37] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E*, vol. 78, no. 4, Article ID 046110, 2008.

[38] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations," *Behavioral Ecology and Sociobiology*, vol. 54, no. 4, pp. 396–405, 2003.

[39] D. E. Knuth, *The Stanford GraphBase: A Platform for Combinatorial Computing*, AcM Press, New York, NY, USA, 1993.

[40] P. M. gleiser and l. danon, "Community structure in Jazz," *Advances in Complex Systems*, vol. 6, no. 4, pp. 565–573, 2003.

[41] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, "The large-scale organization of metabolic networks," *Nature*, vol. 407, no. 6804, pp. 651–654, 2000.

[42] R. Guimera, D. Leon, A. Diaz-Guilera, F. Giralt, and A. Arenas, "Self-similar community structure in a network of human interactions," *Physical Review E*, vol. 68, no. 6, Article ID 065103, 2003.

[43] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 US election," in *Proceedings of the 3rd International Workshop on Link Discovery*, ACM, New York, NY, USA, pp. 36–43, August 2005.

[44] D. Bu, Yi Zhao, L. Cai et al., "Topological structure analysis of the protein-protein interaction network in budding yeast," *Nucleic Acids Research*, vol. 31, no. 9, pp. 2443–2450, 2003.

[45] M. Boguná, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas, "Models of social networks based on social distance attachment," *Physical Review E*, vol. 70, no. 5, Article ID 056122, 2004.

[46] L. N. F. Ana and A. K. Jain, "Robust data clustering," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, Madison, WI, USA, June 2003.