Hindawi
Complexity
Volume 2021, Article ID 1772407, 18 pages
https://doi.org/10.1155/2021/1772407

Research Article

Community Detection Based on Density Peak Clustering Model and Multiple Attribute Decision-Making Strategy TOPSIS

Jianjun Cheng, Xu Wang, Wenshuang Gong, Jun Li, Nuo Chen, and Xiaoyun Chen

School of Information Science and Engineering, Lanzhou University, Lanzhou, Gansu 730000, China

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

Received 23 August 2021; Revised 11 November 2021; Accepted 20 November 2021; Published 7 December 2021

Academic Editor: Giacomo Fiumara

Community detection is one of the key research directions in complex network studies. We propose a community detection algorithm based on a density peak clustering model and multiple attribute decision-making strategy, TOPSIS (Technique for Order Preference by Similarity to an Ideal Solution). First, the two-dimensional dataset, which is transformed from the network by taking the density and distance as the attributes of nodes, is clustered by using the DBSCAN algorithm, and outliers are determined and taken as the key nodes. Then, the initial community frameworks are formed and expanded by adding the most similar node of the community as its new member. In this process, we use TOPSIS to cohesively integrate four kinds of similarities to calculate an index, and use it as a criterion to select the most similar node. Then, we allocate the nonkey nodes that are not covered in the expanded communities. Finally, some communities are merged to obtain a stable partition in two ways. This paper designs some experiments for the algorithm on some real networks and some synthetic networks, and the proposed method is compared with some popular algorithms. The experimental results testify for the effectiveness and show the accuracy of our algorithm.

1. Introduction

Research on complex networks [1] has been an important aspect of data mining. Complex networks are often abstracted from actual systems and are composed of nodes representing entities and edges representing connections between them. Due to the complexity of the mechanism of the systems, the macroscopic behavior of the networks does not conform to the single statistical randomness or the complete regularity, and the networks present a kind of complexity between the two properties. In different applications, the networks present a complex topology structure due to the diversity of the nodes and edges. Researches show that the networks abstracted from the real systems often have such characteristics as small-world [2], scale-free [3], and community structure [4, 5]. The small-world characteristic shows that the nodes in the network are connected by a short path; and the scale-free feature means that the degree of the nodes follows a power-law distribution. The nodes in the network can be divided into several groups, wherein the nodes within each group have more dense connections, and the connections between the groups sparse, with each group constituting a so-called "community." The community structure contains the organizational information in each part of the network and the interaction information between these parts, which can be of great help to the research on the underlying structure and potential functions of the actual systems. And, community detection can promote other complex network studies, like influence maximization [6–8], network embedding [9–11], feature-level fusion [12], and vulnerability assessment [13]. Therefore, finding the community structure has become an important research direction of complex networks.

Community detection can be regarded as a clustering problem in the network. The density peak clustering algorithm [14] argues that the cluster centers have larger density because they are usually surrounded by lots of data points. At the same time, they are separated by the data points, resulting in larger distance between them. That makes them significantly larger in density and distance than the non-center data points. Density peak clustering uses this feature to mine the cluster centers in the dataset and identify clusters of any shape.

The density peak clustering model is highly compatible with the problem of community detection. Due to their scale-free feature, the key nodes of the communities are surrounded by many low-degree nodes, so their densities are relatively higher, and there is larger distance between them because of the sparse connections between the communities. In recent years, there have been many methods for successfully applying the density peak clustering model to address the problem of community detection [15–19], which will be introduced in Section 2.

We find that some traditional methods applying density peak clustering model in community detection often use simple mathematical methods to distinguish the key nodes from the nonkey nodes, which include the product or simple linear combination of nodes' density and distance as the standard of discrimination. These methods often encounter difficulties when distinguishing key nodes whose attributes are not particularly obvious from the node set. The reason is that some nonkey nodes which only have a larger density in the dense community might interfere with the key nodes of the sparser communities due to the intricate structure of the network.

In this paper, we think about this problem from another aspect, and we propose a new DPC-based method of community detection. Because the key nodes have larger density and distance, they become outliers facilitating the use of these properties as the nodes' attributes. Therefore, we can turn the problem into outlier detection in the two-dimensional space about the nodes' properties instead of using traditional simple mathematical methods. The key nodes and other nodes can be distinguished more accurately from this aspect. And, in our method, the DBSCAN algorithm [20] is used for outlier detection.

After obtaining the key nodes, we generate the frameworks of communities by taking them as the seeds, and expand each framework by gradually absorbing the most similar node iteratively. In the process of expansion, we use the T-similarity as a criterion to select the most similar node, and this index is computed by using the multiple attribute decision-making strategy, TOSIS (Technique for Order Preference by Similarity to an Ideal Solution) [21] to combine multiple similarities, and avoids the insufficient adaptation of a single similarity. The expansion phase stops when the benefit of external edges is greater than that of internal edges for the community. Then, we also use the T-similarity to allot the nonkey nodes, which are not covered by the expanded communities, and obtain the initial community structure. At the end of this step, there are some small communities in our partition. To get the final result, we use two strategies to merge some of them, which include an approach by optimizing modularity and a strategy about community metrics.

Optimizing modularity [22] is a fast and quality-assured way of community merge. Therefore, we make use of this strategy to merge some initial communities. Sometimes, this way might merge communities excessively due to the pursuit of modularity, which affects the resolution of communities. Community metric [23] is an indicator to evaluate the significance of a community. So, we also provide a way to solve the resolution problem by controlling the minimum community metric allowed in the network.

The contributions of our work are summarized as follows:

(i) In this paper, we mine the key nodes (cluster centers) in the DPC model from a new perspective that the DBSCAN algorithm is applied to the transformed two-dimensional dataset of network, which provides a new idea for distinguishing the key nodes from the nonkey nodes in the DPC model accurately.

(ii) TOPSIS, a multi-attribute decision-making algorithm, is used to synthesize the four similarities in our method. On the one hand, the communities' expansion and merging operations are more stable and reasonable, and on the other hand, the adaptability of the proposed algorithm to different networks is improved.

(iii) We propose two approaches for merging some of small communities; the difference between them is that they merge the unstable communities in the detection process from different directions. One is inclined to be higher modularity, and the other pays more attention to solve the resolution limit problem. Under the guarantee of the previous steps, either approach will result in a higher quality of partition.

The organization of this paper is as follows. Section 2 investigates the problem of community detection, Section 3 clarifies the specific steps of the proposed method, Section 4 testifies the effect of the proposed method through experiments, and Section 5 concludes the entire paper.

2. Related Work

The research on community detection in complex networks has been ongoing for decades, and a large number of methods have emerged. Here, we introduce some of them that have inspired the ideas of the proposed algorithm.

2.1. Traditional Community Detection Methods. Optimizing modularity is a classical and efficacious way in community detection. And, the modularity is first proposed in the literature [22] along with the GN algorithm, which is used to evaluate the quality of the community structure—larger modularity often means higher quality of the structure. Fast Q [24] is a classic modularity optimization algorithm proposed by Newman. It initially takes each node as a community, and the hierarchical community structure is obtained by continuously merging the two communities with the largest modularity increment. Louvain algorithm [25] obtains the community structure with larger modularity by moving each node to its neighbor's community with the largest modularity increment, then compressing the community as a supernode, and repeatedly performing the above operations until the modularity is the largest. ECG [26] obtains k groups of communities by using one-level Louvain

k times, and weights the edges according to the probability that the two nodes are in the same community, then uses Louvain on the weighted network to get the community partition. Leiden [27] is an improvement of Louvain. The authors argue that weakly connected or even disconnected communities may appear during the operation of Louvain. Therefore, they pay more attention to the connectivity of the communities when moving the nodes. The shortcomings of Louvain are overcome, and the algorithm is also optimized on the time complexity through the fast local node-moving method. In addition, there are many evolutionary algorithms which detect communities aiming at optimizing the modularity [28, 29].

Optimizing modularity is not the only direction for community detection. Based on a variety of information on the network, there have been a lot of methods to partition networks from different directions. Attractor [30] is a method based on network dynamics wherein the interaction between nodes and the distance between nodes affect each other. Under their interplays, there are obvious differences between inter-community edges and intra-community edges on distance. After deleting the inter-community edges with a large distance, the community structure can be obtained. LPA [31] is a community detection algorithm based on the mechanism of information propagation and has high efficiency. Every node is initially assigned with a unique label, and iteratively updates its own label to be the one that occurs in its neighborhood most frequently. The label update procedure is repeated until every node's label is the most frequent one among its neighbors. At that time, the network is divided into communities according to the same labels held by the nodes. Infomap [32] links the problem of community detection with information coding through coding the communities and nodes. When the code length is shorter, the corresponding community structure is clearer. In the process of random walk, Infomap puts the node into its neighbor's community, and takes the community with the largest reduction of code length as the node's affiliation, until the code length reaches the minimum. Walktrap [33] defines the distance between nodes through random walks, and continuously merges the two communities with the shortest distance to obtain multilevel community structure. The structures with the largest modularity are selected as the final partition. PPC [34] is a top-down divisive method, in which an ordered sequence of nodes is generated through random walk repeatedly first, and then PPC uses modularity to determine the cut point to separate subgraphs, and the network is continuously partitioned until there is no positive modularity gain. The literature [35] proposes a community detection method based on game theory. This method defines the leadship of nodes by sequential-move game, and optimizes the attribution of nonleader nodes in dynamic systems. The community partition result with high objective function value is obtained in iterations. In addition, there are many excellent ideas to partition the network from various aspects such as spectral analysis [36–38], nonnegative matrix factorization [17, 39–41], and network embedding [42, 43].

2.2. Community Detection Methods Based on Density Peak Clustering. Since the DPC (Density Peak Clustering) algorithm [14] was proposed in 2014, there have been many methods of applying DPC to detect communities. Isofdp [15] uses IsoMap to map nodes to a d-dimensional space, which presents more diversity while retaining the characteristics of the original network, then calculates the density and distance corresponding to the nodes in the low-dimensional space, and selects k community centers. By repeating the above process in the scopes of d and k, the optimum d and k and the corresponding community structures are obtained. IDPM [16] defines nodes' density and distance using Jaccard similarity and the shortest path length of other nodes, respectively, and mines the key nodes whose density is larger than a calculated threshold; then, the nonkey nodes are assigned to the key nodes correspondingly. Then, IDPM iteratively merges the community and modifies the community affiliations of the unstable nodes in the community boundaries. Finally, the community structure of the network is obtained. IDPCNMF [17] uses the improved page rank score as the density, takes the shortest path length between the node and another one with the larger density as the distance. Then, it calculates the product of the density and the distance, and selects the nodes whose value of product is larger than the mean plus the twice of the standard deviation as the center nodes. Taking the number of center nodes obtained as the parameter of NMF, the adjacency matrix is factorized into vectors containing community structure information; then, IDPCNMF analyzes the communities from the result. EADP [18] weights the link strength through common neighbors, no matter whether there are direct connections between nodes or not. The authors use the reciprocal of the link strength as the distance, and map it to the density using the Gaussian kernel function. This algorithm takes the product of density and distance, calculates the product difference for all adjacent pairs of nodes, and then compares the difference with the value predicted by using a linear regression on other smaller differences. The community centers are obtained automatically. In addition, the expansion phase of EADP has been extended to the scope of the overlapping community detection. CDEP [19] compresses the nodes with degrees of 1 or 2 in the network to higher degree neighbors as super nodes. The degree of each super node is taken as the density, and the number of nodes contained in each super node is taken as the quality. CDEP uses the second-order difference method to mine some super nodes as the seeds, and uses common neighbor weighted similarity as a criterion to expand the seeds to generate communities.

3. Method

This paper proposes a community detection algorithm DPCT, which is based on the density peak clustering model [19] and the multiple attribute decision-making algorithm, TOPSIS [21]. Transforming the network topology into a two-dimensional space related to the density and the distance, the clustering algorithm DBSCAN [20] is used to find the key nodes of the communities. We mine them in a more

precise scope than traditional DPC methods that only use the simple mathematical combination of the density and the distance, so the key nodes are exact and reasonable. Because the networks are different in their topological structure, and single similarity is difficult to reflect the accurate similar situation in different scenarios. Therefore, DPCT uses the multiple attribute decision-making algorithm TOPSIS to calculate a new similarity T-similarity, which can combine the advantages of multiple similarities.

The steps are shown in Figure 1, and can be summarized as follows: (1) Mine the key nodes using DBSCAN on a two-dimensional dataset, which is transformed from the network. (2) Generate the community frameworks and expand them according to the calculated T-similarity. (3) Allocate the remaining nonkey nodes to the existing communities, or generate new small communities on the basis of T-similarity. (4) Merge some of the small communities. The pseudo code of the proposed method is shown in Algorithm 1.

3.1. Key Nodes Mining. The density and distance of the nodes need to be calculated, which makes the network's topological information to be transformed into a two-dimensional space. First of all, the density contribution of a node with degree k to its neighbors is $1/k$, so we use equation (1) to define the density of node v.

$$\rho(v) = \sum_{u \in N(v)} \frac{1}{k(u)}, \tag{1}$$

where $N(v)$ represents the set of neighbors of the node v, and $k(u)$ represents the degree of node u. That is, the density of a node is the sum of all the neighbors' contributions to it. The distance of node v is defined as the minimum length of the shortest path between it and the other nodes with higher density, and the distance of the node with the largest density is the maximum value to the other nodes' distance:

$$d(v) = \begin{cases} \min_{u:\, \rho(u) > \rho(v)} \mathrm{dis}(u, v) & \text{if } \rho(v) < \max\{\rho(x) | x \in V\}, \\ \max_{i,j} \mathrm{dis}(i, j) & \text{if } \rho(v) = \max\{\rho(x) | x \in V\}, \end{cases} \tag{2}$$

where $\mathrm{dis}(u, v)$ refers to the length of the shortest path between the nodes u and v. In order to facilitate the subsequent operations, we normalize the two attributes of each node according to

$$\begin{cases} \rho'(v) = \dfrac{\rho(v)}{\max\{\rho(v) | v \in V\}}, \\ d'(v) = \dfrac{d(v)}{\max\{d(v) | v \in V\}}. \end{cases} \tag{3}$$

In this way, we transform the nodes into a two-dimensional space. Figure 2, for instance, shows the visualization of the dolphin social network [44] and the transformed data points. The key nodes in the four communities are "tigger," "jet," "grin," and "sn96," and they are obvious outliers in the two-dimensional space shown in Figure 2(b). In addition, the node "sn96" is a conspicuous key node in Figure 2(b), but this node does not have a particularly high density due to the sparse connections within its community. Finding the boundary between some key nodes like "sn96" and larger density nonkey nodes is always a problem worth exploring in the application of the DPC model.

In some of them, this boundary is often distinguished by simple mathematical methods. However, we found that some key nodes with lower density value, like the node "sn96," only holds a value of 0.2, and is interfered by nonkey nodes with higher density. Simple mathematical calculation is difficult to prevent such interference, and it may lose some key nodes.

We think about this problem from a more reasonable and accurate aspect, and turn it into an outlier detection problem. According to the scale-free characteristic of the network, the key nodes are outliers in the two-dimensional space, whether their density values are prominent or not. Therefore, we use the DBSCAN to cluster the transformed dataset, and the nodes that cannot be clustered are taken as the key nodes.

DBSCAN is a classic clustering algorithm that clusters the dataset by using two parameters, the radius Eps and the minimal number of data points within the radius Minpts, and the data points that cannot be clustered are outliers. We use this algorithm to find the key nodes from the two-dimensional space like the one presented in Figure 2(b).

We intend to take the outliers which cannot be absorbed in any clusters as the key nodes for community detection. However, some key nodes may be recognized as small clusters on large-scaled networks. In this case, if we only select outliers, some potential key nodes will be left out. If multiple clusters are recognized, there must be some clusters formed by the key nodes. Therefore, we check the attributes of the nodes in each cluster to determine the target clusters, and extract the key nodes from them. It is clearly shown that the nodes in each cluster have similar characteristics. If any member is not suitable for a key node, all the nodes in the cluster cannot be selected. Here, we calculate a density threshold θ as the criterion for examining clusters. The condition for a cluster CL to be excluded is

$$\exists v \in \mathrm{CL}, \quad \rho\prime(v) <\ = \theta. \tag{4}$$

In this paper, the value of θ is set to $\overline{d}/m\rho\prime$, $\overline{d}$ is the average degree of the network, and $m\rho\prime = \max\{\rho'_v | v \in V\}$ is the maximal normalized density. After excluding all the clusters containing the nonkey nodes, the outliers and nodes in the remaining clusters together are selected as the key nodes.

In the above discussion, the largest cluster tends to be formed by nonkey nodes because of the power-law distribution characteristic [3]. Thus, it can be excluded. The pseudo code for the procedure of mining the key nodes is shown in Algorithm 2.

The steps in the procedure are clear. DBSCAN is used for clustering the dataset, which is transformed from the network. After the clusters and the outliers are determined, we examine all the clusters except for the biggest

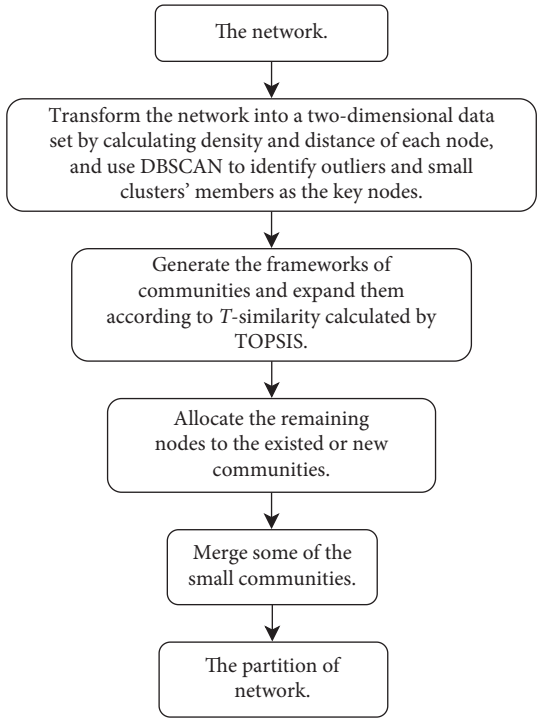


Figure 1: The main procedure of DPCT.

Input: $G(V, E)$, the network; Eps, the radius of the DBSCAN algorithm; Minpts, the minimal number of data points contained within the radius.
Output: partition, the partition of network.
(1) keynodes = Keynodes_mine $(G(V, E), Eps, \text{Minpts})$;
(2) partition ⟵ Community_expansion $(G(V, E), \text{keynodes})$;
(3) nonkeynodes = V-$\{u \mid u \text{ in communities of partition}\}$;
(4) partition = Non-keynodes_allocation $(G(V, E), \text{partition}, \text{nonkeynodes})$;
(5) partition = Commnunity_merge (partition);
(6) **return** partition

Algorithm 1: DPCT, density peak clustering and TOPSIS-based community detection method.

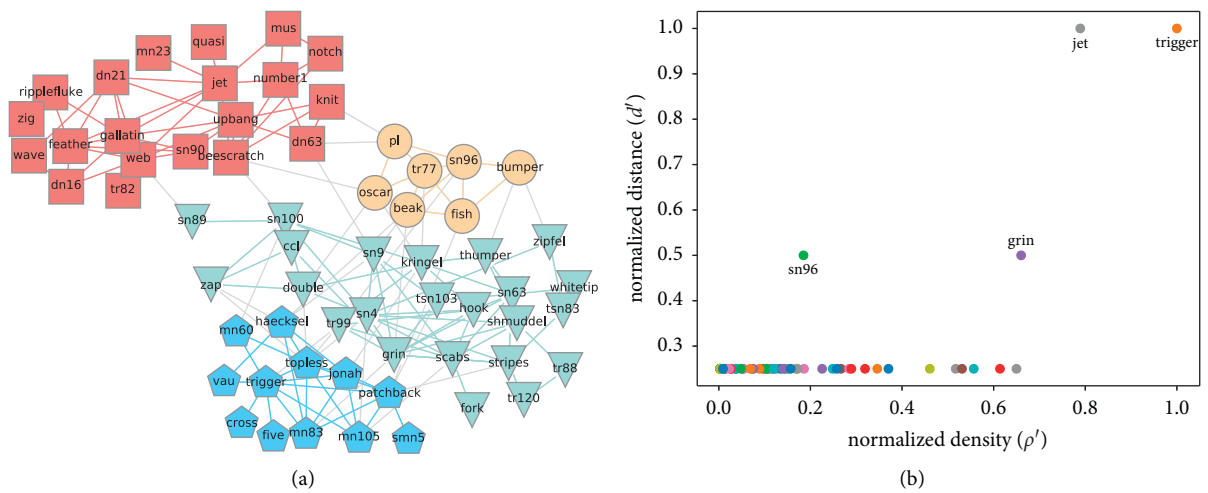


Figure 2: The dolphin network and its transformed data visualization.

Input: $G(V, E)$, the network; Eps, the radius of the DBSCAN algorithm; Minpts, the minimal number of data points contained within the radius.

(1) nodedata$\leftarrow\{(\rho'_v, d'_v)|v \in V\}$;
(2) $\overline{d} = \frac{2}{}(|E|/|V|)$; $m\rho\prime = \max\{\rho'_v|v \in V\}$;
(3) $\theta = \overline{d}/m\rho\prime$;
(4) clusters, outliers = DBSCAN (nodedata, Eps, Minpts);
(5) keynodes = outliers;
(6) $i\leftarrow$argmax$_i\{|C_i||C_i \in$ clusters$\}$;
(7) clusters.remove (C_i);
(8) **for** cluster in cluters **do**
(9) **if** $\exists v \in$ cluster and $\rho(v) < \theta$ **then**
(10) continue;
(11) **else**
(12) keynodes.add (cluster);
(13) **end**
(14) **end**
(15) **return** keynodes

Algorithm 2: Keynodes_mine $(G(V, E), Eps,$ Minpts).

one, and take the members in the suitable clusters and the outliers as the key nodes.

3.2. Community Expansion. After processing using Algorithm 2, we have discovered the key nodes. In this subsection, we generate community frameworks based on them and attach most of the nonkey nodes.

We intend to take each key node as a community initially, but we find that some key nodes hold the close connection. Therefore, they should be contained in the same community. We use the condition presented in equation (5) to determine whether two key nodes v and u are close or not,

$$|N(v) \cap N(u)| > \frac{\min(k(v), k(u))}{2}. \tag{5}$$

That is to say, if the common neighbors between a pair of key nodes are more than half of the smaller degree node's neighbors, they are considered to be closely connected and put together; otherwise, each key node is regarded as an initial community framework separately.

We find that the processing order of these key nodes will affect the expansion result, and the experimental results show that firstly expanding the community framework formed by nodes with larger density and distance has the best effect. Therefore, we integrate the density and the distance of every node $v \in V$ as in the literature [14] to be the product $\gamma(v)$ of them:

$$\gamma(v) = \rho(v) \times d(v). \tag{6}$$

Then, we arrange the community frameworks in the descending order of the largest product of the nodes in each of them.

In the expansion process, there are mainly two problems; one is how to choose the suitable node to add to the community, the other one is the termination condition of the procedure. For the first problem, we attempt to find the most similar node to the community as its new member continuously. We use equation (7) to calculate the similarity between the community and other nodes,

$$s_{\text{avg}}(C, v) = \frac{\sum_{u \in C} s(v, u)}{|C|}, \tag{7}$$

where $s(v, u)$ is the similarity between nodes u and v.

In the experiments, we sometimes found that a single similarity may not adapt to diverse networks. We ponder whether multiple similarities can be combined to integrate the advantages and determine the similar conditions between nodes accurately. So, we propose the T-similarity by introducing the multiple attribute decision-making algorithm TOPSIS to combine multiple similarities that include Jaccard similarity [45], Salton's cosine similarity [46], HPI similarity[47], and HDI similarity [47]. The four similarities between nodes are defined as

$$\begin{cases} s_{\text{Jaccard}}(v, u) = \dfrac{|N(v) \cap N(u)|}{|N(v) \cup N(u)|}, \\ s_{\text{Salton}}(v, u) = \dfrac{|N(v) \cap N(u)|}{\sqrt{|N(v)|} * \sqrt{|N(u)|}}, \\ s_{\text{HPI}}(v, u) = \dfrac{|N(v) \cap N(u)|}{\max(|N(v)|, |N(u)|)}, \\ s_{\text{HDI}}(v, u) = \dfrac{|N(v) \cap N(u)|}{\min(|N(v)|, |N(u)|)}. \end{cases} \tag{8}$$

It can be seen from the above formulae that the most similar node will only appear in the first- or second-order neighbors of the community. Therefore, we only calculate the similarities between the community and the nodes in this area, so that unnecessary calculations can be avoided.

For the issue of the termination condition, since the community is a node group with tight internal connections and sparse external connections, if the number's gain of the

external edges is greater than that for the integral edges after a node is added to the community, this node is not suitable to enter the community. Therefore, if the community expands to a node that satisfies equation (9), the community expansion should be stopped before the node enters.

$$\frac{e_{\text{in}}^{c} - \text{old}e_{\text{in}}^{c} + 1}{\text{old}e_{\text{in}}^{c} + 1} \leq \frac{e_{\text{out}}^{c} - \text{old}e_{\text{out}}^{c} + 1}{\text{old}e_{\text{out}}^{c} + 1}, \tag{9}$$

where e_{in}^{c} and e_{out}^{c} refer to the numbers of internal and external edges after a node is added to the community c, respectively, $\text{old}e_{\text{in}}^{c}$ and $\text{old}e_{\text{out}}^{c}$ refer to the counterparts before the node enters the community. The addition of 1 to the numerator and denominator ensures that the fraction is significant.

The specific steps for community expansion are summarized as the pseudo code in Algorithm 3. We first generate the community frameworks using the single key node or the close pairs of them, and then arrange the frameworks in the descending order of the key node's product of density and distance. Then, T-similarities are calculated, and the most similar node is added to the frameworks continuously until the termination condition presented in equation (9) is met.

3.3. NonKey Nodes Allocation. When the community expansion ends, there are still some nonkey nodes that have not been classified into any community. In order to obtain the community partition of the entire network, these nonkey nodes should be allotted.

To this end, we also use the T-similarity as a criterion to select the most similar node of each unclassified nonkey node. However, some nodes have no connection with their most similar nodes, and putting them into the same community without special consideration may create weakly connected or even disconnected communities. So after finding the most similar node, we classify the nonkey nodes according to the actual situation. For better explanation, we use v to represent the nonkey node to be allocated, and u, C_u to represent v's most similar node and the community to which node u belongs. The different situations and processing strategies are shown in Table 1.

It is clearly shown that the processing order of nonkey nodes also affects the community partition result, and the descending order of the $\gamma(v)$ of each nonkey node v according to equation (6) is optimal distinctly.

The specific steps of this process are as follows. The value of $\gamma(v)$ for each unclassified nonkey node v is calculated firstly, and these nodes are arranged in the descending order of the values. Then, we calculate the T-similarity between each nonkey node and its first- and second-order neighbors; the most similar node is selected, and the nonkey node is allotted according to the specific connection situation of the two nodes. This process is repeated until all nonkey nodes are allotted. The pseudo code of this process is shown in Algorithm 4.

3.4. Community Merging. We have obtained the preliminary partition of the entire network by running Algorithms 2–4, sequentially. However, some preliminary communities each contain only one or two node(s), which are too small so that the intra-community edges are less than the inter-community ones. Here, we merge them using the modularity or community metric as the criteria separately, and the corresponding approaches are named as DPCT–Q and DPCT–M in this paper, respectively.

3.4.1. Community Merging Based on Modularity. Modularity [22] is an important standard to measure the quality of the community partitions, reflecting the connection relationship among the nodes within the communities:

$$\begin{aligned} Q &= \sum_{i=1}^{k} \left(e_{ii} - a_i^2\right) \\ &= \sum_{i=1}^{k} e_{ii} - \sum_{i=1}^{k} a_i^2, \end{aligned} \tag{10}$$

where k represents the number of communities, e_{ij} is the fraction of the number of edges between the communities C_i and C_j to the total number of edges, therefore $\sum_{i=1}^{k} e_{ii}$ represents the proportion of the communities' internal edges, a_i is the sum of e_{ij}, so $\sum_{i=1}^{k} a_i^2$ represents the expectation of $\sum_{i=1}^{k} e_{ii}$. We have mentioned in Section 2 that Fast Q [24] is a classic algorithm targeting at optimizing the modularity. Here, we use the preliminary partition result to replace the initial partition of the original Fast Q algorithm with each single node being a community, and use the modularity increment as the basis for merging the preliminary communities. The modularity increment led by joining a pair of communities C_i and C_j is calculated as

$$\Delta Q_{i,j} = 2\left(e_{ij} - a_i a_j\right). \tag{11}$$

In the merging process, the two communities with the largest modularity increment are joined together each time, and the corresponding modularity increments are updated, until there are no pair of communities that can lead to the positive modularity increment. In the whole process, whether communities are merged or not are determined by the modularity increment, which can merge the small communities in a direction that improves the overall modularity.

3.4.2. Community Merging Based on the Community Metric. Although DPCT–Q has the advantages of parameter-free and large modularity, it might result in the merging of communities excessively and loss of part of the community's information. The community metric [23] measures the significance of a community, and determines which one most needs to be merged. The community metric is used along with the minimum threshold δ; the communities whose community metric is less than δ are merged with the most similar community.

According to the characteristics of community, if a community is small in size and has many connections to the outside, it needs to be merged. Therefore, the community metric for the community C_i is defined as the product of the sparseness α_i and the size fraction β_i of C_i:

Input: $G(V, E)$, the network; keynodes, the keynodes of communities.
Output: partition, the partition of network.

```
partition ⟵ ∅;
for v in keynodes do
    if ∃u ∈ keynodes and |N(v) ∩ N(u)| > min(k(v), k(u))/2 then
        partition.add ({u, v});
    else
        partition.add ({v});
    end
end
arrange community frameworks in partition in the descending order of product of density and distance of nodes in the framework;
for c in partition do
    candidate ⟵ c 's first- and second-order neighbors;
    simmat = calculate the four similarities between c and canidiate;
    while True do
        T-similarity = TOPSIS (simmat);
        simnode ⟵ node with max T-similarity;
        c.add (simnode);
        if e^c_in − olde^c_in + 1/olde^c_in + 1 ≤ e^c_out − olde^c_out + 1/olde^c_out + 1 then
            add simnode's first- and second-order neighbors to candidate;
            update simmat;
        else
            c.remove (simnode);
            break;
        end
    end
end
return partition
```

Algorithm 3: Community_expansion $(G(V, E)$, keynodes).

Table 1: The different situations of nonkey nodes and their most similar node, and the processing strategies.

		$(u, v) \notin E$	$(u, v) \in E$
u is not classified		Generate a new community $\{v\}$	Generate a new community $\{v, u\}$
u is classfied	$\lvert N(v) \cup C_u \rvert > \lvert N(v) \rvert/2$	Add v into C_u	Add v into C_u
	$\lvert N(v) \cup C_u \rvert \leq \lvert N(v) \rvert/2$	Generate a new community $\{v\}$	

Input: $G(V, E)$, the network; partition, the community partition of network; nonkeynodes, the nodes not covered in the expanded communities.
Output: partition, the partition of network.

```
arrange nonkeynodes by γ(v) of each v;
for each v in nonkeynodes do
    candidate ⟵ v's first- and second-order neighbors;
    simmat = calculate the four similarities between u and candidate;
    T-similarity = TOPSIS (simmat);
    u ⟵ node with max T-similarity;
    if (v, u) ∈ E then
        if u in partition then
            add v to the community in which u belongs;
        else
            partition.add ({v, u});
        end
    else
        partition.add (v);
    end
end
return partition
```

Algorithm 4: Nonkeynodes_alloction $(G(V, E)$, partition, nonkeynode).

$$m_i = \alpha_i \times \beta_i, \tag{12}$$

where α_i and β_i are calculated as equations (13) and (14), respectively,

$$\alpha_i = \frac{\left|E_i^{\text{in}}\right|}{\left|E_i^{\text{out}}\right|}, \tag{13}$$

$$\beta_i = \frac{\left|V_i\right|}{|V|}, \tag{14}$$

where E_i^{in} and E_i^{out} represent the internal and external edge sets of community C_i, respectively; V_i represents the set of nodes in community C_i, and V represents that in the entire network. That is, α_i is defined as the ratio of the number of internal edges to the number of external connections for C_i, and β_i is defined as the size of the community relative to the entire network. That means the smaller size of a community and the weaker the internal connections, the more it needs to be merged.

Therefore, we choose the community with the smallest community metric denoted as C_i, then find the community C_j that is most similar to C_i and merge them. The similarity between the two communities is defined as equation (15),

$$\text{Sim}\left(C_i, C_j\right) = \frac{\sum_{u \in C_i, v \in C_j} s(u, v)}{\left|C_j\right|}. \tag{15}$$

Same as the previous discussion, we use the T-similarity for $s(u, v)$ in equation (15).

In this process, we merge the community with the smallest community metric into its most similar community, and update the community metric of the relevant community in turn, until all the communities whose community metric is less than δ are merged.

Comparing the two strategies, it can be seen that DPCT–Q merges communities automatically and tends to acquire a larger modularity. DPCT–M is more inclined to solve the resolution limit problem by adjusting the given threshold.

3.5. Time Complexity. The time complexity of our algorithm is analyzed in this subsection. DPCT can be divided into four phases from the above discussion, and we analyze these steps.

Firstly, we transform the network into a two-dimensional dataset. For each node, the density is calculated in $O(\overline{d})$, where $\overline{d}$ is the mean degree. And, the calculation of distance can be accomplished by a breadth first search process of finding a node with larger density value. According to the small-world law, this node can be found in $O(n)$. Therefore, the time complexity of the transforming step is $O(n^2)$. And, the process of DBSCAN can be accomplished in $O(n \log n)$ [20].

If we obtain k community frameworks, each community finds its most similar node in $O(\log \overline{j})$, and the expansion phase is terminated in $O(\overline{i} \log \overline{j})$, where $\overline{i}$ is the average number of nodes absorbed, $\overline{j}$ is the mean number of communities' neighbors. At the end of the community expansion process, there are u nodes not covered; finding each node's most similar node needs $O(\log \overline{w})$, where $\overline{w}$ is the mean number of nodes' neighbors, and this process can be carried out within $O(u \log \overline{w})$.

In the community merging process, if there are c communities that need to be merged, the modularity optimization way needs $O(c^2)$ [22] and the way of community metric control can be implemented in $O(c \log c)$ [23]. In conclusion, since k, $\overline{i}$, $\overline{j}$, u, $\overline{w}$, c are all much smaller than n, DPCT-Q and DPCT-M partition the network in $O(n^2 + n \log n) \sim O(n^2)$.

4. Experiments

In order to verify the performance of the proposed method, we design these experiments. First, we explore the influence of the DBSCAN's two parameters Eps and Minpts on the quality of the detected community structure, and use the results to set the parameters for the subsequent experiments. After that, we use some mature community detection algorithms to compare the performance with the proposed method on some real and synthetic networks. Finally, we present the comparison between the results of the proposed method with a single similarity and the results of combining four similarities using TOPSIS to testify the advantages of T-similarity.

In our experiments, we use twelve real-world networks and four groups of synthetic networks, which will be introduced in section 4.2. The indexes used are modularity [22] and NMI [48]. The larger the modularity, the better the community structure. The larger the NMI, the closer the detected community structure is to the real structure, and its maximum value is 1.

We choose a variety of comparison algorithms for testifying the proposed algorithm's performance: they are Fast Q [24], ECG [26], Louvain [25], Leiden [27], Attractor [30], Infomap [32], PPC [34], Walktrap [33] and Isofdp [15], respectively. All of them have been introduced in Section 2.

4.1. Experiments on the Settings of Parameters. Because we use the clustering results of DBSCAN, its parameters Eps and Minpts affect the number of the key nodes, so as to affect the final community partition. Before performing the other experiments, we first explore the influence of these two parameters. Here, we conduct experiments on as many values as possible for them on three smaller networks, namely, the karate club network [49], the Riskmap network [50], and the dolphin social network [44], to observe their impact on the quality of the resultant communities. DPCT–M's third parameter δ needs to be adjusted according to the rule explored in the literature [23] after the other two parameters are determined. Therefore, we take the results of DPCT–Q and plot them in the heat maps, as shown in Figures 3–5:

From the figures, we can see that the detected number of key nodes decreases with the decrease of Eps and the increase of Minpts. These findings are logical. First, Eps

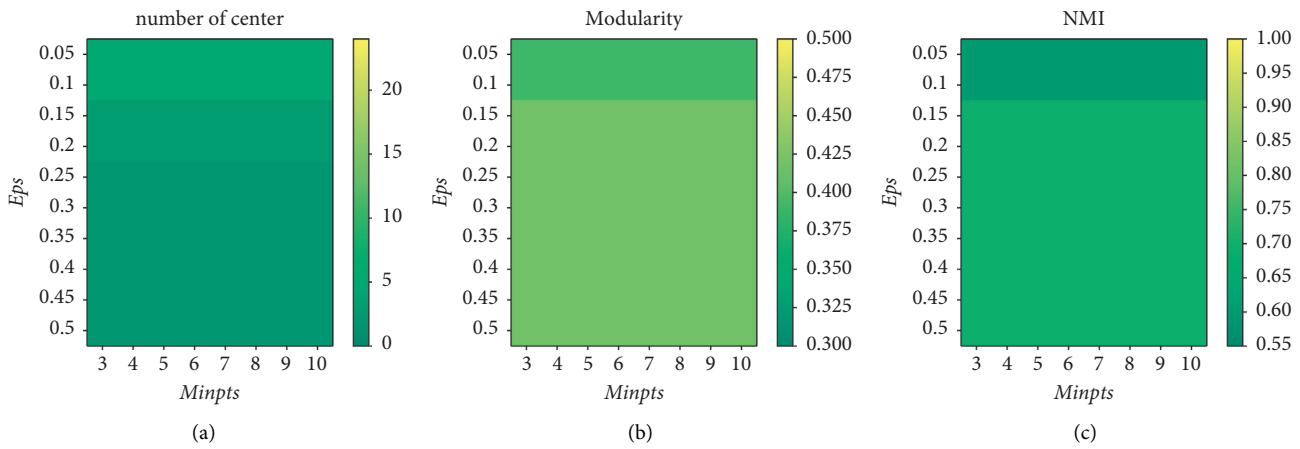


Figure 3: The influence of different parameters on the results in the karate club network. (a) The influence on the number of key nodes. (b) The influence on the modularity. (c) The influence on the NMI. In the figure, the closer the color of blocks to yellow, the greater the value of the block. This illustration style applies to the following figures as well.

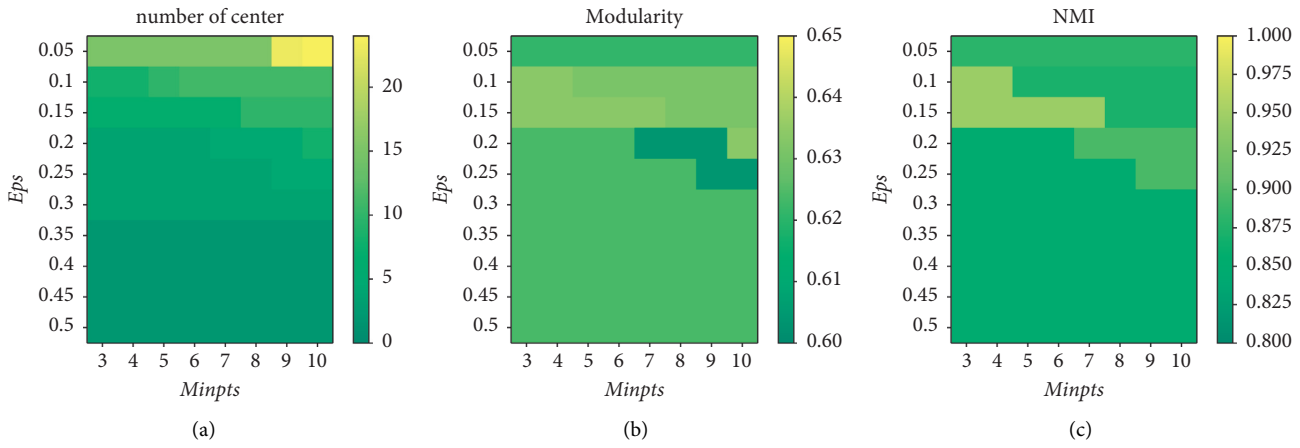


Figure 4: The influence of different parameters on the results in the Riskmap network. (a) The influence on the number of key nodes. (b) The influence on the modularity. (c) The influence on the NMI.

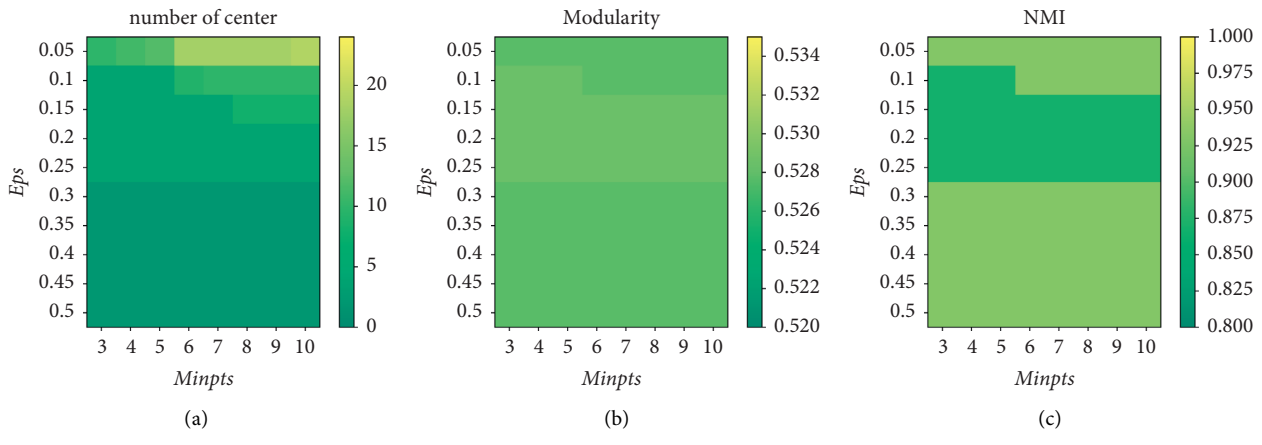


Figure 5: The influence of different parameters on the results in the dolphin network. (a) The influence on the number of key nodes. (b) The influence on the modularity. (c) The influence on the NMI.

represents the radius of the DBSCAN algorithm, and Minpts represents the minimal number of data points contained within the radius. Decreasing Eps or increasing Minpts decrease the probability of data points being clustered. Because we choose the outliers and nodes in the smaller clusters, the number of key nodes will increase as Eps decreased or Minpts increased. However, if these two parameters are modified unfounded to increase the number of key nodes, some inappropriate nodes may be selected, which may lead to a decrease in the performance of DPCT. Although the DBSCAN algorithm is highly sensitive to parameters, the proposed method is robust due to the large difference between the attributes of the key nodes and the nonkey nodes.

From Figures 3–5, we can also find that the best community structures are obtained from these three networks when Eps is 0.1 and Minpts is 3. And, because we normalized the density and distance of nodes, the parameter settings can still refer to this situation on other networks of different scales. The larger the network's scale, the smaller Eps and the larger Minpts need to be tuned.

4.2. Comparative Experiments. In this section, we present the performance of DPCT on 12 real networks and 4 groups of synthetic networks with the comparison algorithms. We use NMI and modularity as the indexes for evaluating the algorithm's performance on the networks with known ground-truth community structures, and use the modularity as the evaluation index for networks with the ground-truth being unknown.

In these experiments, we determine the best Eps and Minpts through DPCT–Q firstly, then apply them to DPCT–M and adjust the parameter δ. For algorithms with parameters such as the Attractor, we adjust the parameters to the best on each network. For algorithms with nondeterministic results, such as Louvain and Leiden, we run each of them 50 times on each network and take the largest value.

4.2.1. Real-World Networks. The information of the real-world networks used is listed in Table 2. For the five networks with real community structures in the table, we visualize the ground-truth structure and the results obtained by DPCT in Figures 6–10.

As shown in Figure 6, both DPCT–Q and DPCT–M split the karate network's two parts into four communities. The difference is that DPCT–M mistakenly assigns the node "10" into the community of node "1." This is because the most similar node for "10" is "29," but there is no edge between the nodes "10" and "29," and the node "10" does not have enough neighbors in the community to which node "29" belongs. Therefore, node "10" is regarded as an isolated community. In the merging process, joining node "10" into the community of node "34" yields larger modularity gain in DPCT–Q. While in DPCT–M,node "10" is more similar with the community of node "1." Compared with the ground truth, both of the two kinds of partitions have larger modularity values.

In Figure 7, DPCT–Q and DPCT–M obtain the same partition from the Riskmap network, and both of them split the community of node "18" in the upper right corner of the ground-truth structure into two smaller communities, because the connections between the two small communities are not close enough.

In the dolphin network, we can see from Figures 8(b) and 8(c) that because we accurately detect the key node "sn96" of the community in the upper right corner, its community has been detected successfully. The nodes "kringer" and "thumper" are mistakenly classified into this community because of the larger modularity or T-similarity. Compared with the ground truth, nodes "sn89," "sn100," "zap," "ccl," and "double" are a tighter group in the community of the node "grin," and they are regarded as a new community by DPCT–Q, and DPCT–M merges this group into the community of the node "trigger".

Figure 9 shows the partition detected on the Santa Fe network. DPCT–Q splits the community of node "7" and the community of node "42" into new communities due to the pursuit of the large modularity. In the partitions of the two methods, node "83" is not correctly classified. This is because it is more similar to the community of node "102".

The football network has more connections than the other four networks. Figures 10(b) and 10(c) show the partitions of the two methods, respectively. Although, both DPCT–Q and DPCT–M merge some communities excessively, which is hard to avoid for some communities with relatively denser inter-community connections, both the partitions have considerable modularity.

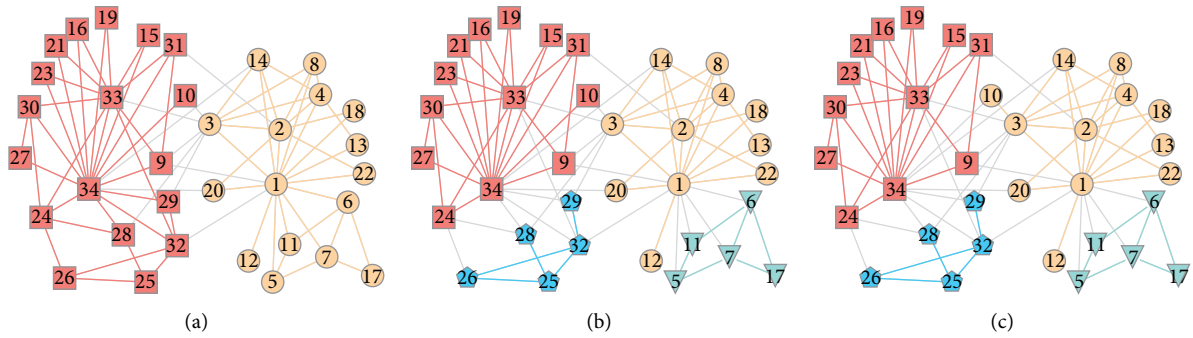
After analyzing the results on these networks, we find that the proposed algorithm can acquire high-quality partitions. In order to better verify the performance of the proposed algorithm, we apply the comparison algorithms on the same networks. Here, we compare the modularity and the NMI of the algorithms' results and present them in the bar charts, as shown in Figure 11. From these figures, we can see that both DPCT–Q and DPCT–M achieve the largest or the second largest modularity. The scenario of NMIs is almost the same as that of modularity, but they are more or less affected by the misclassified nodes. However, the results still have large values on the Riskmap network and the Santa Fe network.

In addition, we test the effect of the comparison algorithms and the proposed algorithm on the other seven real networks, and the results are presented in Figure 12. Due to the large scale of the networks, some comparison algorithms cannot detect the community structure. For example, Attractor, Isofdp, and Walktrap cannot obtain the effective results from the Cond-mat networks. We do not plot the corresponding bars on the bar chart for the algorithms that cannot detect the corresponding results.

In these networks, the community partitions of the proposed algorithm generally have higher quality, which is reflected by the modularity of the results. It can be seen that the proposed algorithm has obtained the largest modularity on the Polbooks network, the e-mail network, the Power network, the PGP network, and the Cond-mat network, and the second largest values on other networks.

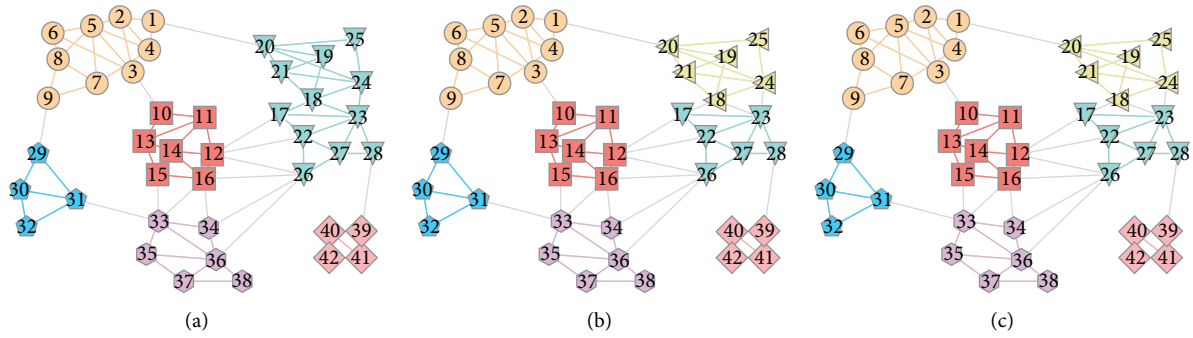
Table 2: The information of real-world networks. (The columns "$|V|$" and "$|E|$" represent the numbers of nodes and edges, respectively. The column "$|C|$" represents the number of communities in the ground-truth structure of the network, a symbol "—" means that the ground-truth community structure is unknown, and *Eps* and Minpts are the parameters.).

Network	$\|V\|$	$\|E\|$	$\|C\|$	*Eps*	Minpts
Karate club [49]	34	78	2	0.1	3
Riskmap [50]	42	83	6	0.1	3
Dolphin [44]	62	159	4	0.1	3
Santa Fe [51]	118	197	6	0.08	3
Football [51]	115	613	12	0.05	3
Polbooks [52]	105	441	—	0.05	3
Les. Mis. [53]	77	253	—	0.05	3
email [54]	1133	5451	—	0.05	3
Facebook [55]	4039	88 234	—	0.05	4
Power [56]	4941	6594	—	0.04	4
PGP [57]	10 680	24 316	—	0.04	4
Cond−mat [58]	27 519	116 181	—	0.03	4



(a) (b) (c)

Figure 6: The karate club network. (a) The ground-truth community structure. (b) The result detected by DPCT–*Q*. (c) The result detected by DPCT–*M*.



(a) (b) (c)

Figure 7: The Riskmap network. (a) The ground-truth community structure. (b) The result detected by DPCT–*Q*. (c) The result detected by DPCT–*M*.

4.2.2. Synthetic Networks. In the experiments on the real networks, the proposed algorithm exhibits excellent performance. Experiments on the artificially synthesized networks also can testify for the proposed algorithm's accuracy. The LFR benchmark networks [59] are a kind of artificially synthesized networks, the characteristics of which are tuned by some parameters. We generate networks with different numbers of nodes, and different community sizes, to meet the needs for testing the algorithm's performance. The main parameters for generating the LFR network are as follows: n represents the number of nodes; k and max$k$ represent the average degree and maximum degree in the network; min$C$ and max$C$ represent the minimum and maximum size of the community in the network; and exp1 and exp2 represent the power-law distribution exponents of the nodes' degree and the size of the communities. In addition, there is the most important parameter μ, which represents the proportion of the edges associated with each node but connecting outside of the node's community. That is, the larger the value of μ, the more difficult it is to detect the community structure.

We generate four groups of networks for experiments, including two scales of networks with 1000 nodes and 5000 nodes; for both scales, we have generated network groups holding small and large communities, which are denoted by

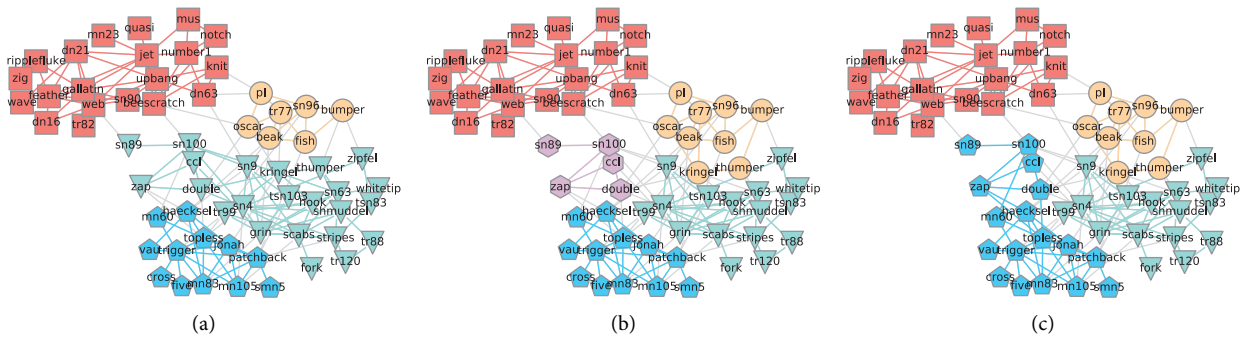


Figure 8: The dolphin network. (a) The ground-truth community structure. (b) The result detected by DPCT–Q. (c) The result detected by DPCT–M.

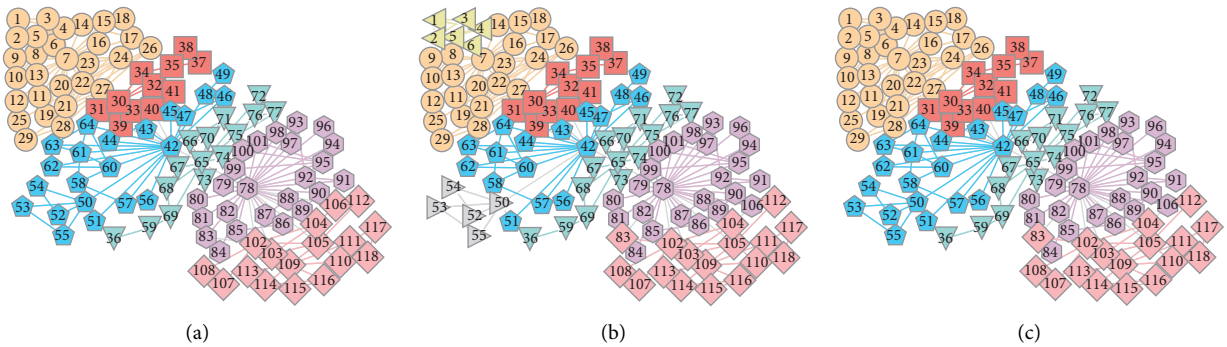


Figure 9: The Santa Fe network. (a) The ground-truth community structure. (b) The result detected by DPCT–Q. (c) The result detected by DPCT–M.

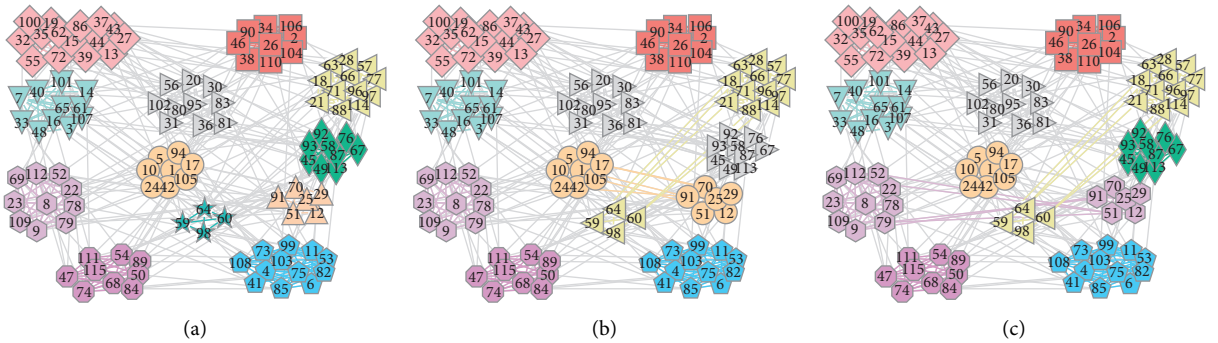


Figure 10: The football network. (a) The ground-truth community structure. (b) The result detected by DPCT–Q. (c) The result identified by DPCT–M.

1000s, 1000b, 5000s, and 5000b, respectively. Each group of networks are generated by varying the values of μ from 0.1 to 0.8 with increasing 0.1 each time, and ten networks are generated in the same parameter setting. For each comparison algorithm, we take the average of the results of the ten networks as its result to minimize the error caused by the occasionality. The specific parameters of the LFR network are shown in Table 3.

In the experiments on the "1000" series of networks, Eps is set to 0.05 and Minpts is set to 3. And, on the "5000" series of networks, Eps is set to 0.04 or 0.03, Minpts is set to 3 as well.

For the convenience of comparison, we draw the results of the algorithms in the line chart, as shown in Figure 13. It can be seen when the value of μ is low, both DPCT–Q and DPCT–M can get considerable NMI. As the value of μ gradually increases, the network's ground-truth structure gradually becomes indistinct. The performance of DPCT–Q which uses modularity increment as the standard is decreasing, and the NMI obtained is relatively low. This phenomenon is also reflected in all the modularity-optimization-based algorithms, and DPCT–Q performs particularly better among them. In contrast, the advantages of DPCT–M gradually appear with the increase of μ. In the four groups of networks, DPCT–M can obtain the best NMI values even when μ is 0.8. These results show that DPCT–M can accurately detect communities even when the community structure is ambiguous.

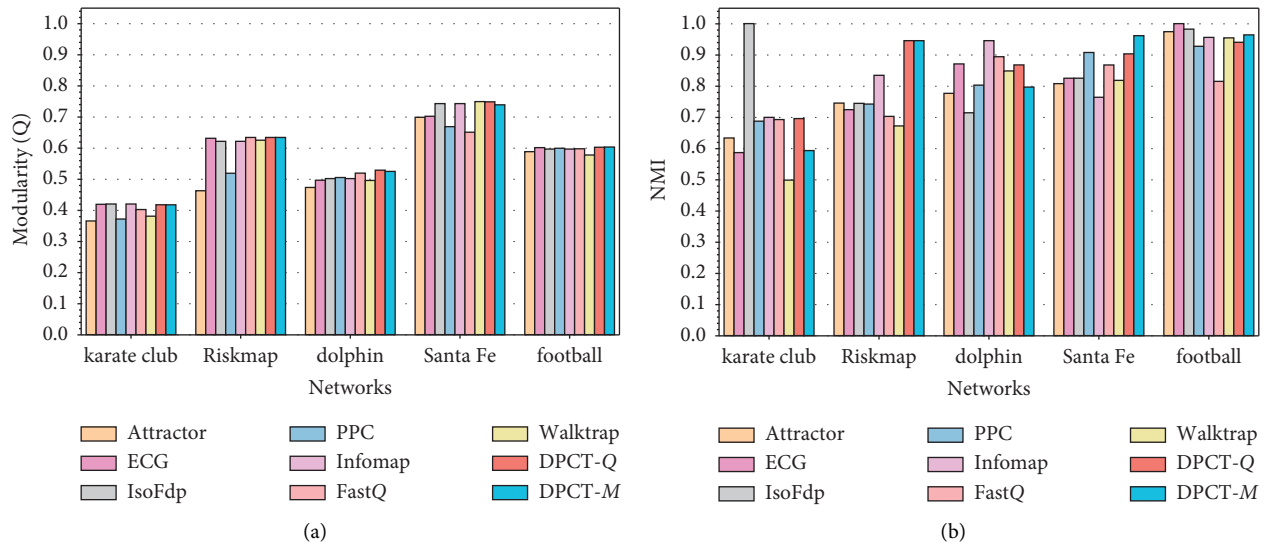


Figure 11: Performance comparison between different community detection algorithms in the networks with ground truth: (a) comparison in modularity; (b) comparison in NMI.

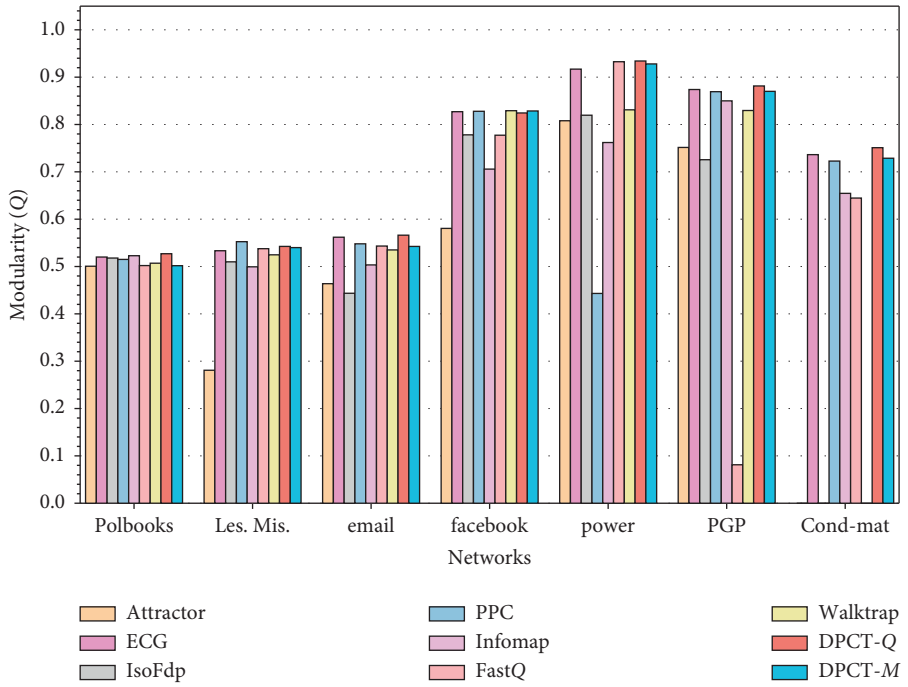


Figure 12: Performance comparison between different community detection algorithms in the networks with ground-truth communities being unknown.

Through comparative experiments, we assess the performance of the proposed algorithm. Irrespective on real networks or on synthetic networks, both DPCT–Q and DPCT–M can obtain higher quality partitions. The accuracy of DPCT–Q decreases in some extreme situations. At the same time, DPCT–M also has a strong ability to adapt to the extreme networks.

4.3. Similarity Analysis Experiments. In our method, we use multiple attributes decision-making algorithm, TOPSIS, to calculate the T-similarity. In this group of experiments, we compare our method with T-similarity and with four different single similarities, respectively, on the five real-world networks with the ground-truth structure. For DPCT–M, we use Jaccard similarity, Salton's cosine similarity, HPI similarity, and HDI similarity in community expansion and nonkey nodes allocation process separately, and the corresponding methods are named as DPCT–Q_j, DPCT–Q_s, DPCT–Q_p, and DPCT–Q_d. For DPCT–M, we also use the four different single similarities to replace T-similarity in the processes; the replaced methods are denoted as

Table 3: The parameters of the LFR benchmark networks, including the number of nodes, the average degree and max degree of nodes, the power-law distribution exponents, the minimum and maximal size of the community, and the mixing parameter μ with the increment of μ, $d\mu$.

Network	n	k	max$k$	exp1	exp2	min$C$	max$C$	$\mu\,(d\mu)$
1000s	1000	20	50	−2	−1	10	50	0.1 ~ 0.8 (0.1)
1000b	1000	20	50	−2	−1	20	100	0.1 ~ 0.8 (0.1)
5000s	5000	20	50	−2	−1	10	50	0.1 ~ 0.8 (0.1)
5000b	5000	20	50	−2	−1	20	100	0.1 ~ 0.8 (0.1)

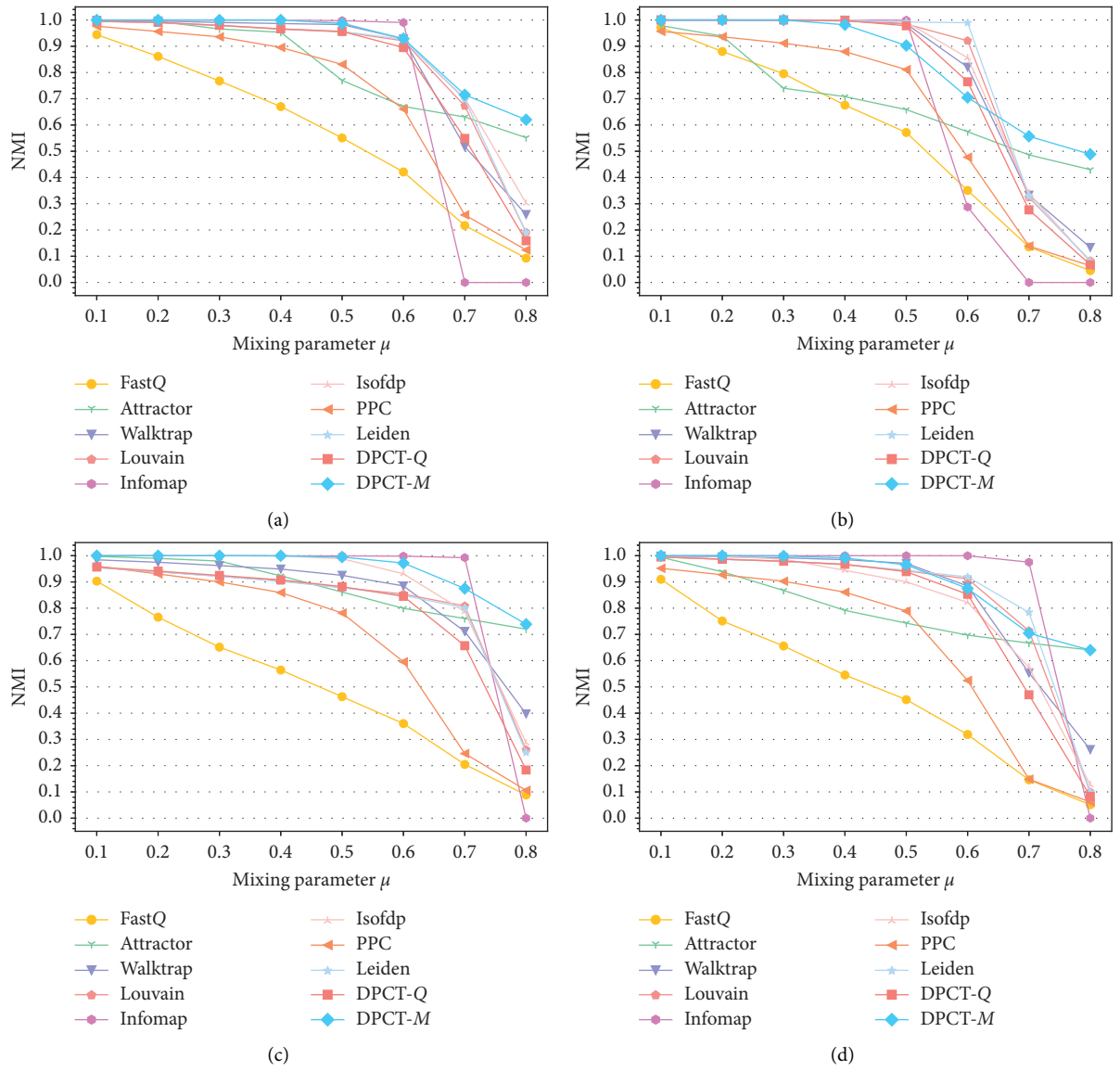


Figure 13: Performance comparison between different community detection algorithms in the LFR networks. (a) The results detected in the 1000s networks. (b) The results obtained from the 1000b networks. (c) The results identified from 5000s networks. (d) The results acquired from 5000b networks.

DPCT–M_j, DPCT–M_s, DPCT–M_p, and DPCT–M_d, respectively. We plot the results in the bar chart for comparison, as shown in Figure 14.

From Figure 14(a), we can see that under the DPCT–Q framework, the modularity obtained on the karate club network by HDI is relatively low, but in contrast it reaches the largest value on the dolphin network. The scenarios for the other three similarities are similar, which testify that the single similarity does not have a good adaptiveness, and the T-similarity can integrate the advantages of four similarities in different networks. There is a similar rule in Figure 14(c). And we can observe that DPCT–Q and DPCT–M with T-similarity can obtain the largest modularity on each network.

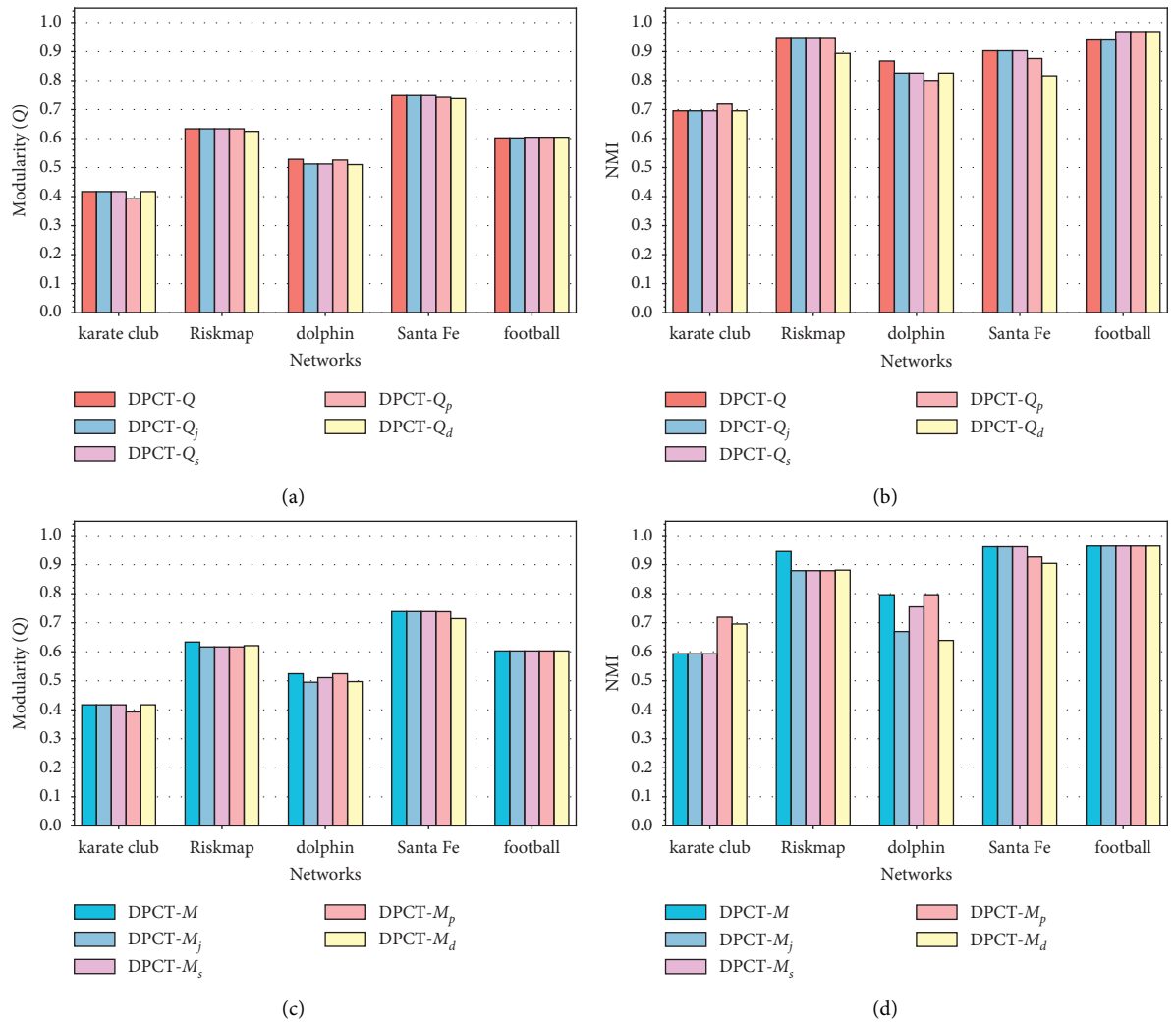


Figure 14: Comparison of the effect of TOPSIS and single similarity. (a) The modularities detected by the DPCT–Q framework with T-Similarity and different single similarities. (b) The NMIs detected by DPCT–Q framework with T-Similarity and different single similarities. (c) The modularities detected by DPCT–M framework with T-Similarity and different single similarities. (d) The NMIs detected by DPCT–M framework with T-Similarity and different single similarities.

These comparison results demonstrate the rationality of the multiple attribute decision-making method. Of course, more similarities can be used as attributes to improve the accuracy. In this paper, four node similarities are selected to improve the quality of the detected communities while taking into account the calculation efficiency simultaneously.

5. Conclusion

This paper presents a community detection algorithm, DPCT, based on the density peak clustering model and multiple attribute decision-making strategy, TOPSIS. Using DBSCAN, we mine the key nodes to avoid interference with the nodes that have larger density, and use TOPSIS to calculate a new well-adapted T-similarity to replace the traditional similarities. The proposed method generates communities' frameworks based on the key nodes and expands them, and allots the remaining nonkey nodes using T-similarity; finally, we use two strategies to merge some of the obtained communities. In the experiments, the influence of the parameters of DBSCAN on the community detection is first explored. After determining the best values of the parameters, the accuracy of DPCT is verified in comparison with other algorithms. Finally, we testify the adaptation of T-similarity.

Two strategies, namely DPCT-Q and DPCT-M, are proposed in this paper, and both of them have their own advantages; the former tends to obtain high-modularity partition and get results more handily, and the latter can partition the networks as accurately as possible even when the community structure is not clear. Therefore, we suggest that DPCT-Q be used to partition the network first in practical applications. On the one hand, the community partition result can be obtained handily, and on the other hand, the suitable parameters Eps and Minpts of DBSCAN can be found. If the required community information is lacking in the result of DPCT–Q, DPCT–M can be used.

Frankly speaking, the time complexity of the proposed method is relatively high, which is mainly due to the fact that the shortest path length between nodes is used as the distance attribute of nodes, which takes more time to acquire the results when it is applied to large-scale networks. Therefore, we will try to use other ways of distance calculation to improve the adaptability of the proposed method in the future work.

Data Availability

The networks used in our experiments include some real-world networks and some artificial datasets. The real-world networks that have been cited in Table 2 were taken from previously reported studies. Most of them can also be downloaded from http://www-personal.umich.edu/~mejn/netdata/ and https://snap.stanford.edu/data/index.html. We construct the Riskmap network manually according to the literature [50]. The artificial networks are synthesized using LFR benchmark network generator, which are freely available at https://sites.google.com/site/santofortunato/, and the parameters used are listed in Table 3.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the Natural Science Foundation of Gansu Province of China (Grant no. 20JR5RA284).

References

[1] S. H. Strogatz, "Exploring complex networks," *Nature*, vol. 410, no. 6825, pp. 268–276, 2001.

[2] D. J. Watts and S. H. Strogatz, "Collective dynamics of "small-world" networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[3] A. L. Barabási, R. Albert, and H. Jeong, "Scale-free characteristics of random networks: the topology of the world-wide web," *Physica A: Statistical Mechanics and Its Applications*, vol. 281, pp. 69–77, 2000.

[4] M. E. Newman and G. Reinert, "Estimating the number of communities in a network," *Physical Review Letters*, vol. 117, Article ID 078301, 2016.

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

[6] J. Shang, S. Zhou, X. Li, L. Liu, and H. Wu, "CoFIM: a community-based framework for influence maximization on large-scale networks," *Knowledge-Based Systems*, vol. 117, pp. 88–100, 2017.

[7] J. Li, T. Cai, K. Deng, X. Wang, T. Sellis, and F. Xia, "Community-diversified influence maximization in social networks," *Information Systems*, vol. 92, Article ID 101522, 2020.

[8] H. J. Li, W. Xu, S. Song, W. X. Wang, and M. Perc, "The dynamics of epidemic spreading on signed networks," *Chaos, Solitons & Fractals*, vol. 151, Article ID 111294, 2021.

[9] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, CA, USA, February 2017.

[10] Y. Li, Y. Wang, T. Zhang, J. Zhang, and Y. Chang, "Learning network embedding with community structural information," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, Macao, China, August 2019.

[11] S. Cavallari, V. W. Zheng, H. Cai, K. C. C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 377–386, Singapore, November 2017.

[12] H. J. Li, Z. Wang, J. Pei, J. Cao, and Y. Shi, "Optimal estimation of low-rank factors via feature level data fusion of multiplex signal systems," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[13] H.-J. Li, L. Wang, Z. Bu, J. Cao, and Y. Shi, "Measuring the network vulnerability based on markov criticality," *ACM Transactions on Knowledge Discovery from Data*, vol. 16, no. 2, pp. 1–24, 2021.

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

[15] 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.

[16] Z. H. Deng, H. H. Qiao, M. Y. Gao, Q. Song, and L. Gao, "Complex network community detection method by improved density peaks model," *Physica A: Statistical Mechanics and Its Applications*, vol. 526, Article ID 121070, 2019.

[17] H. Lu, Z. Shen, X. Sang, Q. Zhao, and J. Lu, "Community detection method using improved density peak clustering and nonnegative matrix factorization," *Neurocomputing*, vol. 415, pp. 247–257, 2020.

[18] M. Xu, Y. Li, R. Li, F. Zou, and X. Gu, "EADP: an extended adaptive density peaks clustering for overlapping community detection in social networks," *Neurocomputing*, vol. 337, pp. 287–302, 2019.

[19] X. Zhao, J. Liang, and J. Wang, "A community detection algorithm based on graph compression for large-scale social networks," *Information Sciences*, vol. 551, pp. 358–372, 2021.

[20] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining Kdd*, pp. 226–231, Portland Oregon, August 1996.

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

[22] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 69, Article ID 026113, 2004.

[23] J. Cheng, X. Su, H. Yang et al., "Neighbor similarity based agglomerative method for community detection in networks," *Complexity*, vol. 2019, Article ID 8292485, 16 pages, 2019.

[24] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, vol. 69, Article ID 066133, 2004.

[25] 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.

[26] V. Poulin and F. Théberge, "Ensemble clustering for graphs," in *Proceedings of the International Conference on Complex*

Networks and Their Applications, pp. 231–243, Springer, Cambridge, UK, December 2018.

[27] V. A. Traag, L. Waltman, and N. J. E. Van, "From Louvain to Leiden: guaranteeing well-connected communities," *Scientific Reports*, vol. 9, pp. 1–12, 2019.

[28] C. Pizzuti, "GA-net: a genetic algorithm for community detection in social networks," in *Proceedings of the International Conference on Parallel Problem Solving from Nature*, pp. 1081–1090, Springer, Dortmund, Germany, September 2008.

[29] K. Chen and W. Bi, "A new genetic algorithm for community detection using matrix representation method," *Physica A: Statistical Mechanics and Its Applications*, vol. 535, Article ID 122259, 2019.

[30] 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*, pp. 1075–1084, Sydney, Australia, August 2015.

[31] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 76, Article ID 036106, 2007.

[32] M. Rosvall and C. T. Bergstrom, "Maps of information flow reveal community structure in complex networks," 2007, https://arxiv.org/abs/0707.0609.

[33] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *Proceedings of the International Symposium on Computer and Information Sciences*, pp. 284–293, Springer, Istanbul, Turkey, October 2005.

[34] S. A. Tabrizi, A. Shakery, M. Asadpour, M. Abbasi, and M. A. Tavallaie, "Personalized pagerank clustering: a graph clustering algorithm based on random walks," *Physica A: Statistical Mechanics and Its Applications*, vol. 392, no. 22, pp. 5772–5785, 2013.

[35] H. J. Li, Z. Bu, Z. Wang, and J. Cao, "Dynamical clustering in electronic commerce systems via optimization and leadership expansion," *IEEE Transactions on Industrial Informatics*, vol. 16, pp. 5327–5334, 2019.

[36] L. Donetti and M. A. Muñoz, "Detecting network communities: a new systematic and efficient algorithm," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2004, no. 10, Article ID P10012, 2004.

[37] L. Ni, P. ManMan, J. Wenjun, and L. Kenli, "A community detection algorithm based on multi-similarity method," *Cluster Computing*, vol. 22, no. S2, pp. 2865–2874, 2019.

[38] P. Shi, K. He, D. Bindel, and J. E. Hopcroft, "Local lanczos spectral approximation for community detection," in *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 651–667, Springer, Skopje, Macedonia, September 2017.

[39] J. Yang and J. Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach," in *Proceedings of the sixth ACM international conference on Web search and data mining*, pp. 587–596, Rome, Italy, February 2013.

[40] W. Wu, S. Kwong, Y. Zhou, Y. Jia, and W. Gao, "Nonnegative matrix factorization with mixed hypergraph regularization for community detection," *Information Sciences*, vol. 435, pp. 263–281, 2018.

[41] H.-J. Li, L. Wang, Y. Zhang, and M. Perc, "Optimization of identifiability for efficient community detection," *New Journal of Physics*, vol. 22, no. 6, Article ID 063035, 2020.

[42] D. Jin, X. You, W. Li et al., "Incorporating network embedding into Markov random field for better community detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 160–167, Honolulu, HI, USA, January 2019.

[43] F. Liu, S. Xue, J. Wu et al., "Deep learning for community detection: progress, challenges and opportunities," 2020, https://arxiv.org/abs/2005.08225.

[44] 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.

[45] P. Jaccard, "Étude comparative de la distribution florale dans une portion des Alpes et des Jura," *Bulletin de la Société Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.

[46] G. Salton, *Modern Information Retrieval 1983*, McGraw-Hill, New York, NY, USA, 1983.

[47] A. L. Barabási, E. Ravasz, and Z. Oltvai, "Hierarchical organization of modularity in complex networks," in *Statistical Mechanics of Complex Networks*, pp. 46–65, Springer, Berlin, Germany, 2003.

[48] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and experiment*, vol. 2005, no. 9, Article ID P09008, 2005.

[49] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, no. 4, pp. 452–473, 1977.

[50] K. Steinhaeuser and N. V. Chawla, "Identifying and evaluating community structure in complex networks," *Pattern Recognition Letters*, vol. 31, no. 5, pp. 413–421, 2010.

[51] 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.

[52] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[53] D. E. Knuth, *The Stanford Graphbase: A Platform For Combinatorial Computing*, AcM Press, New York, NY, USA, 1993.

[54] R. Guimerà, L. Danon, A. G. Díaz, F. Giralt, and A. Arenas, "Self-similar community structure in a network of human interactions," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 68, Article ID 065103, 2003.

[55] J. J. McAuley and J. Leskovec, "Learning to discover social circles in ego networks," *NIPS Citeseer*, vol. 2012, pp. 548–556, 2012.

[56] R. Albert, I. Albert, and G. L. Nakarado, "Structural vulnerability of the North American power grid," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 69, Article ID 025103, 2004.

[57] M. Boguñá, R. S. Pastor, A. G. Díaz, and A. Arenas, "Models of social networks based on social distance attachment," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 70, Article ID 056122, 2004.

[58] M. E. Newman, "Clustering and preferential attachment in growing networks," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 64, Article ID 025102, 2001.

[59] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 78, Article ID 046110, 2008.