A Multi-label Propagation Algorithm for Community Detection Based on Average Mutual Information

Yinan Chen, Dong Li, and Meng Ye

South China University of Technology, Guangzhou 510006, China

Correspondence should be addressed to Dong Li; cslidong@scut.edu.cn

Received 24 February 2022; Accepted 29 March 2022; Published 9 May 2022

Academic Editor: Kuruva Lakshman

Copyright © 2022 Yinan Chen et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Community structure is one of the vital characteristics of complex networks. How to effectively detect communities is a hot issue. From the perspective of information theory, the community structure of complex networks can be detected and revealed more accurately. This research introduces the average mutual information (AMI) into the detection process of the multi-label propagation algorithm (MLPA) and proposes a new community detection algorithm AMI-MLPA. The algorithm initially determines the propagation order according to the influence of nodes in the network. By selecting the label with stronger propagation intensity and smaller conditional entropy in the process of label propagation, a more reasonable community partition can be obtained. Experiments on real-world datasets and synthetic datasets show that the algorithm is better than FastGN, GN, and other LPA-based algorithms in general, with high accuracy of results on large-scale synthetic networks, which verifies the effectiveness of the algorithm.

1. Introduction

Communities are groups of strong-related nodes that can be considered as independent compartments of a network. The study of community structure helps analyze the relationship between individuals and group characteristics in complex networks, thereby revealing the structural characteristics and evolution laws of complex systems such as including social, biological, and technological networks [1, 2]. Community detection is the process of revealing community structure from complex networks. Currently, it is applied to the extraction of social groups based on interests, work, friends, and so on in social networks and also the prediction and control of key infection sources and transmission routes in infectious diseases networks. Detecting communities on different varieties of networks proves to be a hard task [2]. Aiming at revealing the community structure of a network, a good community detection algorithm should be able to detect different kinds of network communities accurately, quickly, and stably. This paper mainly focuses on the label-propagation-based community detection algorithms and proposes a novel algorithm named AMI-MLPA by introducing the average mutual information.

Unlike the typical community detection algorithm that uses community evaluation indexes to optimize partitions, label propagation algorithms (LPAs) simulate the evolution of the communities by imitating the communication and assimilation among nodes. It has the advantages of simple logic and high speed. The original LPA is first proposed by Raghavan et al. in 2007 [3], but it can only be applied to nonoverlapping community structures and has apparent randomness. Dai et al. put forward the MLPA [4] in 2013 by changing the form of propagation from single-label to multi-label and replacing random propagation by using propagation intensity to determine the candidate labels. The improved algorithm has higher accuracy and can be applied to overlapping community detection. However, there are significant differences between the results of MLPA with the ground-truth communities on several classic real-world datasets. In addition, it is difficult to select the appropriate parameters since there is obvious difference among the results obtained with different parameters.
In consideration of MLPA’s advantages and shortcomings, our research focuses on improving and optimizing MLPA with the aid of average mutual information (AMI). As an essential basis for quantitative research on information flow, mutual information can be used as an overall measure of information also in the evolution process of communities. When communities are generated, divided, merged, or extinct, AMI can be used to judge the stability between the states before and after the transformation. Guided by this idea, the propagation of labels is controlled toward a larger AMI value in the algorithm, to get a more stable result which is also closer to the real-world community structure.

The main contributions of this paper are summarized as follows:

(i) A new strategy is proposed based on the node influence model, which determines the update order and label selection while propagating, to improve the stability of label propagation.

(ii) By introducing average mutual information into MLPA, a new algorithm AMI-MLPA is proposed, which proves to be more accurate and reasonable.

(iii) Experiments are carried out to verify the accuracy and stability of the algorithm on both real-world and synthetic datasets.

2. Related Work

2.1. Community Detection. In recent years, researchers have proposed numerous types of community detection algorithms, including modularity optimization, spectral clustering, dynamic analysis, clique percolation, and other techniques [2]. The classical modularity-based GN [5] and FN [6] methods can already achieve community detection for most small networks. However, the former is very slow with a time complexity of $O(n^3)$, while the latter improves the efficiency of the former at the expense of the accuracy of the algorithm.

By defining network microscale measurements, such as node similarity, researchers try to extend the idea of clustering for community detection. Zhang et al. proposed CDRS [7] based on defining the similarities between nodes and communities. The similarities determine the lower and upper approximation node sets for each community and then constantly adjust the core nodes and boundaries of the communities until convergence using the idea of K-means. However, the algorithm does not give a convincing central node selection strategy. Using a random method, degree centrality, betweenness centrality, and proximity centrality as central node selection strategies have significantly different performances on different datasets, and it is difficult to determine which choice is the best. With a similar idea to CDRS, SAG-Cluster [8] detects communities by using a collaborative similarity combining structural similarity and context similarity, which are used to, respectively, measure the link-based distance and attribute-based distance of nodes in the network. Nodes are then clustered using K-medoids. However, the two similarity measurements are not completely unified in one framework. Besides, how to find a balance between structure and properties, i.e., how to find a suitable $\alpha$ value for a specific network, is also a tricky problem.

In addition to directly defining the similarities between nodes and communities, network representation learning (NRL) can also be used to transform the community detection problem into a clustering problem, such as in [9, 10]. By using NRL, nodes are converted into vectors, so that the similarities between vectors and between clusters can be calculated with traditional methods, such as Euclidean distance and Ward’s distance. However, the shortcoming of NRL-based methods is that it is challenging to ensure that the structural information of the graph is not lost in converting a network into vectors. In particular, a large number of NRL methods have many hyperparameters, and there is no clear consensus on the selection of suitable hyperparameters.

Researchers found that for a community, as an internally close connected part in a network, it can exist independently of other parts, and its structural characteristics can also be measured in a local approach. Therefore, local community detection methods have been proposed one after another to solve the inefficient problem for detecting communities in large networks. LFM [11] and GCE [12] are typical local community detection algorithms based on a local community evaluation index called fitness. These methods follow the idea of starting from a few nodes or clusters in the network and gradually expanding the communities to the neighborhood until reaching the maximum value of the community evaluation indexes. In recent years, by introducing heuristic methods, the efficiency and stability of local search in local community detection are enhanced, such as using the swarm intelligence optimization framework in LSSO/CD [13]. The local methods can greatly improve the efficiency of community detection through parallel computing, but they inevitably involve three issues: first, how to define a good local community evaluation index; second, how to make full use of global information; and third, how to handle the edge nodes, overlapping community nodes, and orphan nodes. For the three problems, different algorithms have different solutions, but the accuracy of local methods is often lower than that of global methods.

2.2. Label Propagation. Most of the community detection approaches consist of two parts: the first is the community evaluation index, which measures the strength and weakness of a community structure; the second is the community transformer, which constantly changes the community structure and tries to find a partition that reaches some peak values of the evaluation index, that is, the solutions of community detection. Unfortunately, since there is no accurate definition of the merits of community structure, it is hard to design a perfect community evaluation index which is able to evaluate the goodness of the community structure precisely [10].

Label propagation algorithms (LPAs) sidestep this problem. They detected communities following these steps: initially, each node is given an independent label, then a set
of rules is established to spread labels among nodes, and finally, the nodes with the same label are allocated into the same community. In the whole process, no community evaluation index is needed [14]. This enables LPAs efficiently detect communities based on such a simple logic.

Many researchers have been trying to extend LPA to get high accuracy and performance in community detection. Gregory proposed COPRA [15] to extend single-label propagation to multiple labels for the first time, by assigning a belonging coefficient to the labels while propagating. The hyperparameter \( \nu \) of COPRA specifies the maximum number of labels that a node can receive from its neighbors. This means that when there are more than \( \nu \) optional labels, some labels will be randomly discarded during the propagation, resulting in instability of the algorithm. Based on COPRA, several algorithms have been proposed. SLPA [16] maintains a list of labels for each node. During each propagation, the neighbor nodes (speakers) of the current node (listener) decide which labels to send according to the probabilities of the labels in its list, while the listener stores all received labels in its own list. The algorithm analyzes the label lists of all nodes in the final stage and then discards labels with low frequency. This reduces the possibility of information loss during the propagation process. MLPA [4] also maintains a label list for nodes and calculates the propagation intensities of labels during the propagation process. Propagation intensities reflect the structural information and previous propagation information. Only those labels with propagation intensities lower than the threshold will be discarded to reducing information loss. Both SLPA and MLPA can solve the problem that COPRA loses a lot of information during the propagation procedure, but due to the random initialization method and uncertain propagation order, the result of them is still unstable. By a preprocessing step throughout the target network, BMLPA [17] finds rough cores of the network and starts label propagation from them, which further increases the stability of label propagation and the accuracy of the results. Also following the idea of finding core node clusters to start label propagation, the DCN [18] algorithm uses density peaks [19] method to find core nodes. The labels of nodes directly connected to the core nodes are initialized to be the same as the core nodes. This method performs well on datasets with a small number of communities and can obtain very accurate partitions. However, it is debatable to directly classify the adjacent nodes of the core nodes into the same community as the core nodes, because some bridge nodes may directly connect to the cores of communities, without belonging to them. This leads to the problem that on a network with a large number of communities and a small community size, such as the Football dataset, DCN cannot accurately reveal the community structure. Instead of finding the core of the communities to start label propagation, GLPA [20] adopts a reverse thinking that uses label propagation to find closely linked clusters in the network. These clusters are treated as a supernode, and then, a community evaluation metric called merging factor is adopted to merge them into communities further. However, the problem of the approach is that the members of the supernodes cannot be modified during the second merging process. Once there is an error in the process to obtain densely linked clusters using label propagation, it will have a significant impact on the merging step. GLLPA [21] uses dynamic weights to reduce the information loss and improve the accuracy in the propagation process, making the algorithm more accurate than other LPA-based algorithms. The personal influence and the neighboring influence are defined to calculate the dynamic weight, and the alpha hyperparameter is used to control the balance between them. However, experiments show that the introduction of this balancing factor has no significant effect on the accuracy of the algorithm but makes the process of calculating dynamic weights more computationally expensive.

2.3. Network Preprocessing. Aiming at improving the accuracy of the results and reducing the convergence time while searching optimal partitions, recent research is trying to add some preprocessing steps before community transforming to collect more structural information from the network. For example, the community detection method based on positive/negative connections [22] runs a random walking process in the network and performs statistical analysis on the random-walking sequence. Then, the relationships of the nodes are evaluated as positive/negative for further detection. With a similar idea, the EdMot algorithm [23] and LCD-Motif [24] use motif-based hypergraphs of the target network to enhance the edges. The former applies other state-of-the-art algorithms to partition the enhanced network, and the latter detects communities by seeking a sparse vector in the span of the local spectra. The DEMON method [25] builds an Ego-Minus-Ego network from the original network, by combining the ego network extraction and the graph-vertex difference operation. Also, based on ego network, SONIC-MAN [26] uses moderator nodes to integrate the local structural information in distributed online social networks. Such preprocessing steps are verified to be effective in improving the community detection performance of existing approaches.

Node influence analysis is also used as a method of network preprocessing. As another important attribute of complex networks, node influence is closely related to community structure [27, 28]. Currently, there are many methods to evaluate the influence of nodes, which can be divided into three categories: topology-based, content-based, and behavior-based. For network topology diagrams composed only of nodes and edges, evaluation criteria are generally based on topological structure characteristics, such as centrality indexes [29], local clustering coefficient [30], and density peaks [19]. The centrality indexes mainly include degree centrality, betweenness centrality, and closeness centrality, of which degree centrality is a local index, and the latter two are global indexes [29]. Degree centrality measures the influence of a node by measuring the degree of each node and its neighbors [31]. However, the process of converting the adjacency information into degrees loses a lot of structural information, which often leads to large errors in judging the influence of nodes. Betweenness centrality is defined as the number of shortest paths passes through a node between all pairs of
nodes in the network. Closeness centrality measures the volume of walks a node takes to reach all other nodes. Both of them are global indexes measuring node influence accurately but have the significant disadvantage of high computational consumption, especially for large-scale networks. Local clustering coefficient is defined as the ratio of the number of connected edges between nodes to the maximum possible edges that can be connected between nodes in the neighborhood of a node. However, researchers found that the clustering coefficient may not be suitable for measuring the influence of nodes. Clusters with high clustering coefficients may even inhibit the spread of information [32].

Density peaks approach [19] analyze and find the core nodes in the network by calculating the density $\rho$ of nodes and the distance $\delta$ between nodes and their nearest high-density nodes. These two properties can be used to measure node influence. However, it has defects such as difficulty in selecting hyperparameters and low efficiency.

Inspired by the above researches, our research focuses on the improvement of LPA-based community detection from the perspective of information theory. As stated above, MLPA improves the stability of label propagation by introducing multiple labels and propagation intensity. However, the result partition of MLPA has a large difference from the ground truth on several classical datasets. This paper focuses on the optimization and improvement of MLPA, by introducing information entropy and average mutual information. We also calculate the influence of nodes based on node similarity, as a preprocessing step of our algorithm, to increase its stability and accuracy.

3. The Influence of Nodes

3.1. Problems Existing in Label Propagation. There is a problem pressing for a solution in the existing label propagation algorithm. In the process of label propagation, a node determines its label in the previous iteration, before it begins to propagate its label to other nodes. This naturally leads to a key question: whether the order of propagation will have a great impact on the result?

As mentioned above in Section 2, an obvious drawback of LPA is the randomness of its results. Given a particular network, LPA may output various of community partitions even with the same parameter set. For example, the communities detected by LPA on Karate network [33] can have significant differences when running several times. As shown in Figure 1, the two partitions are apparently different on either the number of communities or the ownership of nodes. To make matters worse, LPA brings much more possible outcomes on larger networks, which is not desirable for practices that require unique and stable results.

The primary cause for such a difference is the updating order. There are two alternative update strategies in implementation: random order or following the node indexes. The former usually leads to great randomness in results. The latter produces stable results while it is not rigorous, because the node indexes can only be regarded as nominal attributes, but not ordinal or numerical attributes. If the labels are propagated in accordance with the indexes of nodes, the equality and balance among nodes will be destroyed in the process of propagation.

Another reason that causes the instability results is, when updating a node with an equal frequency of different labels within the neighborhood, a random label is selected. Figure 2 shows an example of such a situation:

The network has a simple symmetric structure with a bridge which is an edge between nodes 3 and 4. While first updating node 3, as the number of labels of its neighbors $a$, $b$, and $d$ are equal, a random label will be selected among them. If $a$ is select, then the label of node 2 will be changed to $a$ when it is updated. The propagation of labels can enter a stable 2-community structure if the same procedure is done to the right half of the network, as shown on the left path in Figure 2. But if label $d$ is selected while updating node 3, and it is then propagated to node 2, the network may enter another stable situation that all nodes have the same label, as shown on the right path in Figure 2.

In these two cases, the algorithm converges to different results. This is because more than one labels share an equal frequency in the neighborhood of the first updating node. Furthermore, the node has a critical position in the network. The early updating of such nodes will significantly affect the process of propagation in LPA and leads to randomness in the results.

3.2. Node Influence Quantification Model. As illustrated in the example above, the community structure detected by LPA is not only decided by the network structure but also affected by the order in which nodes are updated and the selection of adjacent node labels. The main reason for the uncertainty of the result is that different nodes in the network usually have different structural conditions. Updating the labels of some nodes may have more impact on the community structure than that of other nodes. To solve this problem, we introduce the node influence quantification model, as a prestep of LPA, which can determine the label selection according to the structural feature of the network.

Node influence is a concept used to measure the importance of a node in the network, which is proportional to the correlation between this node and the other nodes. Greater node influence stands for greater importance the node is in the network and stronger relationships with the other nodes.

Given a Network $G$, the influence of node $v$ is denoted as follows:

$$\text{Inf}_{G}(v).$$

(1)

Most methods take the definition of node similarity as the premise, and node influence is essentially the accumulation of node similarity. Node similarity is an important evaluation criterion used to measure the relevance between two nodes in a network. As the basis of node clustering, the greater the similarity between two nodes, the more significant the possible correlation between them.

Node similarity can be generally defined in two ways: based on attributes and based on links. Attribute-based computation requires the attribute information of the nodes. By constructing the attribute vectors, nodes are mapped to n-
dimensional space, to calculate the pairwise similarity by Euclidean distance or cosine distance. The link-based method requires the structural information of the nodes, i.e., the link relationship between nodes. There are two kinds of node similarity analysis based on links: global and local. Global link-based approaches involve a large amount of computation, which are mainly concerned with the shortest path length between nodes, the number of independent paths between nodes, etc. [34, 35]. The results obtained by such approaches are often accurate, but it is high time consumption on large-scale networks. Local approaches only gather the structural information within the neighborhood for each nodes, which are faster, but at the expense of accuracy.

In this paper, local link-based node similarity is used to design the node influence model. This is based on the following considerations: First, node similarity can be used not only to calculate node influence, but also to get the propagation intensity of labels in the subsequent label propagation procedure. This will be explained in detail in Section 4. Second, our research objects are network topologies without additional information such as attributes, so link-based methods are more suitable compared with the others. Finally, we hope to improve the accuracy and stability of LPA while keeping its high efficiency, so a local method of measurement is selected to avoid high computational costs.

The following functions are considered to calculate node influence:
(i) **Func. 1.** Node influence is equal to the sum of similarities between a node and its neighbors:

\[ \text{Inf}_G(v) = \sum_{v' \in \Gamma(v)} \text{Sim}(v, v') \quad (2) \]

(ii) **Func. 2.** Node influence is equal to the product of similarities between a node and its neighbors:

\[ \text{Inf}_G(v) = \prod_{v' \in \Gamma(v)} \text{Sim}(v, v') \quad (3) \]

(iii) **Func. 3.** Node influence is equal to the arithmetic average of similarities between a node and its neighbor nodes:

\[ \text{Inf}_G(v) = \frac{\sum_{v' \in \Gamma(v)} \text{Sim}(v, v')}{|\Gamma(v)|} \quad (4) \]

\( \Gamma(v) \) in these equations denotes the set of neighbor nodes of node \( v \).

The form of sum in Func.1 considers two situations in which a node should have a higher influence: one is that the node has more neighbors; the other is that the node is more inclined to associate with other nodes. This implies that the greater the sum of similarity between a node and other neighbor nodes, the greater the possibility of information flowing to other nodes through this node.

Func.2 is roughly the same as that of Func.1, except that it uses a product form instead of sum, which may increase the difference of influence among nodes. From the perspective of algorithm implementation, this approach is much more computationally expensive and memory-intensive than the first, leading to higher costs in both time and space on a large data set.

Func.3 uses the mean value of node similarities between a node and its neighbors, to eliminate the average effect within the neighborhood. A higher node influence based on Func.3 indicates that the node is more inclined to associate with other nodes.

After the node influence function is determined, the definition of node similarity is still needed. As described in the previous section, a local link-based similarity is considered to reduce calculation costs. Table 1 shows the most frequently used indexes to measure the similarity of nodes based on local links. For data sets with different types of network structures, more accurate results can be obtained when a more appropriate node similarity index is selected.

\( \text{Sim}(v, u) \) in Table 1 denotes the similarity between nodes \( v \) and \( u \).

### 3.3. Node Influence Extended LPA

To determine which of the functions is more suitable for LPA, we carry out experiments on Karate network [33], a small network with two communities consisting of 16 and 18 nodes, respectively. The following steps are added to LPA:

First of all, before the propagation begins, the update order of nodes is determined based on node influence.

Secondly, in the process of propagation, nodes with low influence values will be updated first. As illustrated in Figure 2, if a node with a high influence value is updated first, the whole network will probably converge to one community in just a few iterations.

In addition, if multiple labels share the equal frequency for the neighborhood of a node, then the label with the greatest node influence will be selected to propagate.

The three node influence functions are, respectively, added to LPA and run 10 times on Karate network, compared with the original LPA. The number of communities (CN) and the size of each community (CS) of the results are shown in the following table:

As demonstrated in Table 2, the results of the original LPA are significant instability, while extended with node influence, and the other three algorithms give deterministic results. When Func.1 is adopted as the calculation formula of node influence, the number of communities is consistent with the ground-truth of the dataset. When using Func.2, all nodes are allocated to one large community. This is due to the fact that the product form amplifies the influence of some nodes, which leads to a situation that labels tend to propagate to the whole network through these high influence nodes. Such a result makes little sense for community detection tasks. When Func.3 is used to extend LPA, the number of communities in the results is more than that labeled in the dataset, i.e., labels are impeded while propagating within the community. The average form greatly reduces the influence of high-degree nodes, which, however, happen to be the core of a community.

Although the LPA extended by Func.1 has the same number of communities in its results as labeled in the dataset, there are still 6 incorrectly allocated nodes. This is caused by the characteristics of LPA: only the label with the largest number will be accepted while updating, discarding all other labels. If misjudgments occur on some key nodes, the error will continue accumulating in the subsequent iterations and lead to significant errors in the final results. In the next section, we will introduce multi-labels.

### Table 1: Similarity indexes based on local information.

| Name      | Definition                                                                 |
|-----------|-----------------------------------------------------------------------------|
| Salton(Cosine) | \( \text{Sim}(v, u) = \frac{|\Gamma(v) \cap \Gamma(u)|}{\sqrt{|\Gamma(v)| \cdot |\Gamma(u)|}} \) |
| Jaccard   | \( \text{Sim}(v, u) = \frac{|\Gamma(v) \cap \Gamma(u)|}{|\Gamma(v) \cup \Gamma(u)|} \) |
| HPI       | \( \text{Sim}(v, u) = \frac{|\Gamma(v) \cap \Gamma(u)|}{\min \{ |\Gamma(v)|, |\Gamma(u)| \}} \) |
| HDI       | \( \text{Sim}(v, u) = \frac{|\Gamma(v) \cap \Gamma(u)|}{\max \{ |\Gamma(v)|, |\Gamma(u)| \}} \) |
| PA        | \( \text{Sim}(v, u) = |\Gamma(v)| \cdot |\Gamma(u)| \) |
and average mutual information to improve this kind of defect.

4. Multi-label Propagation Based on Average Mutual Information

4.1. Average Mutual Information.

Average mutual information (AMI) comes from the information theory, which is a quantity to measure the amount of information that one random variable contains about the other. In the process of community evolution, the AMI value ($I_p$) can be used to evaluate the stability of communities when they evolve. By calculating the $I_p$ between each pair of adjacent states, the states with larger $I_p$ can be found as the better community partitions [36]. $I_p$ can be calculated by [37]:

$$I_p(X; Y) = \sum_{X_i, Y_j \in X \times Y} P(X_i, Y_j) \log \frac{P(X_i, Y_j)}{P(X_i)P(Y_j)},$$

where $X$ and $Y$, respectively, denote the community partition before and after the communities transform. $X_i$ and $Y_j$, respectively, denote the $i$-th community in partition $X$ and the $j$-th community in partition $Y$.

In order to apply AMI to multi-label propagation, the following two problems need to be solved:

(1) In the process of label propagation, each node does not belong to a specific community, but has a list of community labels with corresponding propagation intensities. It is more suitable to regard such a state as an overlapping partition. How to calculate the probability distributions $P(X_i)$, $P(Y_j)$, and $P(X_i \cap Y_j)$ for the corresponding overlapping communities in such a state?

(2) When a node is updating its labels from neighbors, how to figure out which of these labels makes a larger AMI value before and after the update?

For problem (1), in a nonoverlapping partition, the probability distribution of community $X_i$ can be calculated by the following:

$$P(X_i) = \frac{|X_i|}{N},$$

where $|X_i|$ denotes the number of nodes in community $X_i$, $N$ denotes the total number of nodes of the network.

In the same way, the joint probability of two communities $P(X_i, Y_j)$ can be calculated by the following:

$$P(X_i, Y_j) = \frac{|X_i \cap Y_j|}{N},$$

where $|X_i \cap Y_j|$ denotes the number of nodes that both in community $X_i$ and $Y_j$.

In a nonoverlapping partition, the values of $|X_i|$ and $|X_i \cap Y_j|$ can be obtained by simply counting the number of nodes in the communities. But in the process of multi-label propagation, each node has a list of propagation intensities, to indicate how inclined they are to be included in the corresponding community. Such a list can be represented by a vector $c_v$ with $C$ dimensions, where $C$ stands for the number of communities in the network:

$$c_v = [c_v(1), c_v(2), \ldots, c_v(C)].$$

The value of each dimension $c_v(i)$ denotes the propagation intensity to community $i$ for node $v$.

For each node, the sum of all propagation intensities for the corresponding communities is 1:

$$\sum_i c_v(i) = 1.$$

Thus, the propagation intensity $c_v(i)$ can be used to represent the probability for node $v$ to select community $i$:

$$P_v(X_i) = c_v(i).$$

Then, the probability distribution of community $X_i$ in an overlapping partition can be calculated by summing up the

| CN | Original | Func.1_EXT' | Func.1_EXT | Func.1_EXT |
|----|----------|-------------|-------------|-------------|
| 1  | 3        | 11, 5, 18   | 3           | 34          |
| 2  | 2        | 15, 19      | 10, 24      | 3           |
| 3  | 2        | 17, 17      | 1           | 5, 5, 24    |
| 4  | 1        | 34          | 1           | 5           |
| 5  | 4        | 11, 2, 3, 18| 1           | 5, 5, 24    |
| 6  | 2        | 18, 16      | 1           | 5           |
| 7  | 3        | 15, 13, 16  | 1           | 5           |
| 8  | 2        | 16, 18      | 1           | 5           |
| 9  | 2        | 17, 17      | 1           | 5           |
| 10 | 3        | 14, 3, 17   | 1           | 5           |
probabilities of selecting each node $v$ in the network multiplied by the probability for node $v$ to select community $X_i$: 

$$P(X_i) = \sum_{v \in V} p(v) \cdot P_v(X_i) = \sum_{v \in V} \frac{1}{N} \cdot c_v(i) = \sum_{v \in V} \frac{c_v(i)}{N},$$  

(11)

where $V$ denotes all nodes of the network. 

Here for each node $v$, it is not necessary to specifically distinguish whether it is in community $i$. If the node does not belong to community $i$ at all, the corresponding propagation intensity for the community is 0. It does not affect the sum. 

Equation (11) can be also used to calculate $P(Y_j)$. 

For the joint probability distribution $P(X_i, Y_j)$, in non-overlapping communities, it is essentially the number of nodes in the overlap part of two communities, divided by the total number of nodes in the network. However, in overlapping communities, the overlapping part is not completely determined. Even if two communities $X_i$ and $Y_j$ overlap on node $v$, as node $v$ does not completely belong to community $X_i$, it needs to be multiplied by the intensity corresponding to community $X_i$. It is the same for $Y_j$. Thus, the joint probability for two communities $X_i$ and $Y_j$ can be calculated by the following: 

$$P(X_i, Y_j) = \sum_{v \in V} p(v) \cdot P_v(X_i) \cdot P_v(Y_j) = \sum_{v \in V} \frac{c_v(i) \cdot c_v(j)}{N}.$$  

(12)

For problem (2), the following strategy is used to evaluate which of the labels brings a larger AMI value: 

1. The current partition for evaluation will be represented as $X_{v_0}$. 
2. For each nonzero intensity $C_v(i)$ of node $v$, create a new partition $X_{v'}$ by changing the intensity to 1, and the intensities for other communities to 0. 
3. Calculate the AMI between $X_{v_0}$ and $X_{v'}$ for each label $i$, i.e., $I_p(X_{v_0} : X_{v'})$. A larger AMI value implies that the corresponding label is more appropriate for node $v$. 

Considering that we only need to know which label of node $v$ has a larger $I_p(X_{v_0} : X_{v'})$, the following $\Delta I_p$ is defined to reduce the computation: 

$$\Delta I_p(v_0, v_1) = I_p(X_{v_0} : X_{v_1}) - I_p(X_{v_0} : X_{v_1}) = \sum_{X_v, X_{v'}} P(X_v, X_{v'}) \log \frac{P(X_v, X_{v'})}{P(X_v)P(X_{v'})} - \sum_{X_v, Y_{v'}, X_{v''}} P(X_v, Y_{v'}) \log \frac{P(X_v, Y_{v'})}{P(X_v)P(Y_{v'})}.$$  

(13)

It can be deduced from Equation (13) that, if a pair of communities $X_{v_1}$ and $X_{v_2}$ remains unchanged in partition $X_{v_0}$ and $X_{v_1}$, they will be cancelled out in $\Delta I_p$. Hence, only the communities with a change on their intensities need to participate in the calculation. For each label of node $v$, step (3) in the above strategy can be replaced by calculating $\Delta I_p$ for each of the labels, and a smaller $\Delta I_p$ value implies that the corresponding label is more appropriate for node $v$. 

4.2. The Algorithm. In a nutshell, the idea of AMI-MLPA algorithm is as follows: firstly, the update order of nodes is determined by the node influence model. Next, according to the propagation intensity and average mutual information, nodes start to update their labels from their neighbors. Finally, the iteration stops when the label of each node converges to a steady state. Figure 3 shows the main procedure of the algorithm. 

Specifically, the algorithm will be described below in three parts: preprocessing, label propagation, and termination of iteration. 

4.2.1. Preprocessing. In this part, each node will be initially allocated into a unique community according to its index. For example, node 1 will be allocated into community 1. This is implemented by assigning a propagation intensity vector $c_r$ for each node $r$. The value of the $r$-th dimension of the vector $c_r(r)$ is set to 1.0 and the others 0. Then, these nodes are sorted ascending according to their node influence, which can be calculated by similarity indexes based on local link information as shown in Table 1, together with Equation (2). 

4.2.2. Label Propagation. According to the order obtained in preprocessing part, each node is updated in multiple iterations following these steps: 

1. Each node $r$ will receive a label $l_t$ from each of its neighbors $t$ (which is the label with the highest intensity in their respective propagation intensity vectors), together with its intensity $c_r(l_t)$ 
2. For each label $l_t$ received, calculate a new propagation intensity $c_r(l_t)'$ using the following equation: 

$$c_r(l_t)' = \sqrt{\text{Sim}(r, t) \cdot c_r(l_t)}.$$  

(14)

3. where $\text{Sim}(r, t)$ is the similarity between nodes $r$ and $t$. If two or more identical labels are received, they will be combined into one with their propagation intensities added up. 

4. The propagation intensity for each label in $c_r$ will be then normalized, as shown in Equation (15), to construct a new propagation intensity vector: 

$$\text{Sim}(r, t) = \frac{\sum_{v \in V} p(v) \cdot P(v) \cdot P(v) \cdot P(v)}{\sum_{v \in V} p(v) \cdot P(v) \cdot P(v) \cdot P(v)}.$$  

(15)
For each nonzero intensity label $l_r$ of node $r$, calculate its corresponding $\Delta I_p$ using Equation (13).

Filter the labels. The parameter $p$ is used to control the maximum number of labels of each node. As $\Delta I_p$ is smaller, the overall average mutual information value is larger. Therefore, the first $p$ labels with smaller $\Delta I_p$ will be preserved, while the intensities of others will be set to 0.

The intensity for each label in $c_r$ will be normalized again to make sure that the sum of all propagation intensities of a node is 1.

The above steps are iterated over several times until the termination condition is met.

4.2.3. Termination of Iteration. After several iterations of the label propagation process, the community partition of the network tends to be stable, where the number of types of labels in the network remains unchanged after an iteration. After the final iteration of label propagation, each node has $p$ labels with nonzero intensities, which means one node is allocated into $p$ communities, which is unreasonable in most cases. At this point, the final filter is applied to preserve as few labels (usually one or two) as possible that best fit each node.

The final filter uses parameter $d$ to determine which of the labels should be preserved. As shown in Equation (16), propagation intensity $c_r(l)$ for label $l$ of node $r$ will be set to 0 if not satisfied:

\[
c_r(l) \geq \max (c_r) - d,
\]

where $\max (c_r)$ represents the maximum of propagation intensities in $c_r$ of node $r$. It can be inferred that if $d = 0$, nonoverlapping communities are generated, because only the labels with the highest propagation intensity are...
The reasonable range of $d$ is $[0, 1]$. The larger $d$ is, the larger the overlapping part will be in the result partition.

The following pseudocode gives a brief description of AMI-MLPA:

4.3. Time Complexity Analysis. The time complexity of this algorithm can be calculated in three parts:

(1) Preprocessing. The time complexity of this part is mainly reflected in the calculation of node influence. Since the calculation of node influence is actually the sum of node similarity with its neighbors, the average time complexity is $O(Nk)$, where $N$ is the number of nodes in the network and $k$ is the average degree of nodes.

(2) Label Propagation. The label initialization will have a time complexity of $O(N)$. As shown in the pseudocode, the label propagation step is the most time-consuming part of the algorithm. Firstly a $T$-times iteration is needed, where $T$ is much smaller than the scale of the network. For instance, it is usually about 10 in an LFR synthetic network with 2000 nodes in our experiments. In each iteration, $p$ labels of $N$ nodes are processed, including updating of nodes and the calculation of the conditional entropy for each label. The updating of nodes will have a constant time complexity for each neighbor of each node; therefore, the time complexity is $O(TNkp)$, where $k$ is the average degree of the nodes. The calculation of $\Delta l_p$ will have a higher time complexity. Since the joint probability for each pair of communities should be calculated, all labels for all nodes have to be calculated in $O(TNC^2p)$, where $C$ is the number of communities. The overall time complexity of the label propagation part is $O(TNC^2kp)$.

(3) Iteration Stop. The final filter needs a comparison for the intensity of each label, which brings a time complexity of $O(Np)$.

During the actual running of the algorithm, $T$ and $k$ are much smaller than the number of nodes $N$ and the number of communities $C$, and $p$ is a constant parameter; thus, the
The overall time complexity of AMI-MLPA is approximate $O(NC^2)$.

## 5. Experiments and Analysis

### 5.1. Preparation of the Experiments

This section covers the baselines and benchmarks of the experiment, aiming at verifying the effectiveness of AMI-MLPA, on both nonoverlapping and overlapping community detection tasks. For nonoverlapping community detection, LPA-based algorithms are mainly used for comparison, including the original LPA [3], MLPA [4], DCN [18], GLPA [20], and GLLPA [21]. In addition, classical algorithms and other state-of-art algorithms are also included for a supplement, including GN [5], FastGN [6], and CDRS [7].

Two types of datasets are used to conduct the experiment, real-world and synthetic. Since these datasets contain ground-truth labels for nodes, the number of communities ($CN$) and normalized mutual information index (NMI) are used to measure the similarity between experimental results and ground-truth community partitions.

The NMI between the results of the algorithms and the known partitions can be calculated using Equation (17) [38]:

$$\text{NMI} = \frac{-2\sum_{i=1}^{C_A} \sum_{j=1}^{C_B} C_{ij} \log_2 \left( \frac{C_{ij}N}{C_iC_j} \right)}{\sum_{i=1}^{C_A} C_i \log_2 \left( C_i/N \right) + \sum_{j=1}^{C_B} C_j \log_2 \left( C_j/N \right)},$$

where $A$ and $B$ represent the two partitions of the network, respectively, $C_{ij}$ is a confusion matrix, the element of which $C_{ij}$ represents the number of nodes both in community $i$ of $A$ and community $j$ of $B$. $C_A$ and $C_B$ represent the number of communities in partitions $A$ and $B$, respectively. $C_i$ and $C_j$ are, respectively, the sum of the $i$-th row and $j$-th column elements in the confusion matrix $i$. $N$ is the total number of nodes of the network.

For overlapping community detection, the accuracy of AMI-MLPA is compared with overlapping community detection algorithms Ego-splitting, DEMON, and BigClam, on both real-world and synthetic datasets. We use overlapping modularity ($Q_{ov}$) [39], extended modularity (EQ) [40], and average conductance (AC) to evaluate the structural merits of the result partitions, instead of comparing the result partition with the labeled networks. Although we did not use the partition labels directly, we still compared the result CN with the labeled CN, because the number of communities still has reference value on overlapping community networks.

### Table 4: Parameters of AMI-MLPA

| Dataset   | Similarity | $p$ |
|-----------|------------|----|
| Karate    | PA         | 3  |
| Dolphins  | PA         | 3  |
| Football  | Jaccard    |    |
| Polbooks  | PA         | 6  |
The extended modularity (EQ) is a community quality index which extends the definition of Newman’s modularity to overlapping community structures. The definition of EQ is shown as follows:

\[
EQ = \frac{1}{2m} \sum_{c \in C} \sum_{i,j \in c} \frac{1}{O_i O_j} \left[ A_{ij} - \frac{k_i k_j}{2m} \right],
\]

where \( m \) is the total number of edges in the network, \( c \) is one of the communities, \( i \) and \( j \) are nodes belong to community \( c \); \( k_i \) and \( k_j \) are their degrees, respectively. \( A_{ij} \) is the element of the adjacency matrix which follows the condition that when nodes \( i \) and \( j \) are linked, the value of \( A_{ij} \) is 1, otherwise 0. \( O_i \) and \( O_j \) are the numbers of communities to which nodes \( i \) and \( j \) belong and give EQ the ability to deal with the case that a node belongs to more than one community. A high value of EQ indicates a significant overlapping community structure for a particular network.

The overlapping modularity index \( Q_{ov} \) is another quality function proposed by Nicosia et al. [39] to extend modularity to the more general case of overlapping communities, defined as follows:

\[
Q_{ov} = \frac{1}{m} \sum_{c \in C} \sum_{i,j \in V} \beta_{(i,j),c} A_{ij} - \frac{P_{out}^{out} k^i_{out} P_{in}^{in} k^i_{in}}{m} \]

Same as that in EQ, \( m \) is the total number of edges in the network. \( A_{ij} \) are the elements of the adjacency matrix. \( i \) and \( j \) are nodes of the network. \( k^i_{out} \) is the out-degree of node \( i \),
while $k_j$ is the in-degree of node $j$. Instead of the number of communities to which a node belongs to, $Q_{ov}$ uses the belonging coefficients $\beta_{l(i,j)} \text{ and } \beta_{l'(i,j)}$ to calculate the weight for each link $l(i,j)$ existing between nodes $i$ and $j$. Alike to EQ, a higher value of $Q_{ov}$ indicates a stronger community structure.

Another index used in our experiment to evaluate the quality of overlapping community detection result is the average conductance (AC). Conductance is a local measure for the goodness of a node cluster in the network [41]. For a cluster $c$, conductance is defined as follows:

$$f(c) = \frac{m^{\text{out}}_c}{2m^{\text{out}}_c + m^{\text{in}}_c},$$  \hspace{1cm} (20)$$

where $m^{\text{in}}_c$ is the number of internal edges of cluster $c$ and $m^{\text{out}}_c$ is the number of edges that links the cluster to other parts of the network. To evaluate a partition of the network, average conductance (AC) is used, which is defined as follows:

$$\Phi(C) = \text{avg} \ f(c),$$  \hspace{1cm} (21)$$

where $C$ is a community partition of a network. The equation averages the conductance of all communities of the partition.

The hardware and system information of the experiments are shown in Table 3.

5.2. Parameter Analysis

5.2.1. Maximum Label Capacity. Considering that the value of $p$ may have an impact on the results of community detection, we analyzed the parameter $p$ on the Football network [5]. Figure 4 shows the community partition results of AMI-MLPA and the NMI with the labeled values when $p$ ranges from 2 to 10, respectively.

It shows that if $p$ is too small for the network (e.g., $p = 2$ for Football), it will have a negative effect on the outcome since valid label information is discarded in the process of propagation. When $p$ reaches a particular value (e.g., $p \geq 4$ for Football), the results tend to be stable with high accuracy.
5.2.2. Overlapping Control Parameter. The overlapping control parameter $d$ determines the number of overlapping nodes of the communities detected. It can be referred from Equation (16) that a label will be reserved in the final filter if its propagation intensity is large enough compared with the maximum of the node. If $d = 0$, only the labels with the highest propagation intensity are retained. If $d = 1$, since propagation intensity must be in $[0, 1]$, then max ($c_i$) $- d \leq 0$; hence, none of the labels will be filtered. Note that the number of labels for a node at this time is not only related to $d$, but also to $p$. That is because $d$ is only used when filtering labels in the last step, but in the previous label propagation process, $p$ determines how many labels a node can retain at most.

Figure 5 shows $Q_{ov}$ of the results applying different parameter $d$ on the Karate network [33]. It shows that when $d = 0.15$, $Q_{ov}$ reaches its highest value on Karate. If $d$ is too large, it may lead to unreasonable overlap between communities, resulting in a decrease in $Q_{ov}$, even lower than the value of generating a nonoverlapping partition. The appropriate value of $d$ is usually between 0.1 and 0.15.

5.3. Nonoverlapping Community Detection. To verify the results of AMI-MLPA, 4 real-world datasets are adopted including Karate [33], Dolphins [42], Football [5], and Polbooks [43]. Experiments show that the results of AMI-MLPA are very close to the known partitions of these datasets, which is better than other traditional algorithms. The parameters and results are shown in Tables 4 and 5 and Figure 6, where the bold numbers emphasize the best experimental results within each dataset. Note that $d$ is set to 0 for nonoverlapping communities.

The results show that AMI-MLPA performs well on real-world networks. The numbers of communities of the networks are correctly figured out by AMI-MLPA. Although the NMI of it does not get the highest NMI on every dataset, but the average accuracy is the best compared with the other algorithms.

The synthetic network datasets are created using GN-128 network [5] and LFR networks [38] with different attributes including network size, average degree ($k$), and mixed parameter ($\mu$), as shown in Table 6:

The results of AMI-MLPA on these synthetic networks are shown in Table 7. As illustrated in Table 7, the algorithm correctly figured out the number of communities on almost all experimental networks. The NMIs between the result and the labels partition are at about 0.97 to 1.0, even when the mixing parameter ($\mu$) is set to 0.5 in LFR-1000_3, which means the community structure in such network is highly obscure. Experimental results show that on GN-128 and LFR synthetic networks, the community detection results of AMI-MLPA are highly accurate regardless of the network size, node degrees, and the number of communities.

5.4. Overlapping Community Detection. To verify the effectiveness of the algorithm on overlapping community detection tasks, we set the parameter $d$ of AMI-MLPA to 0.1 to find overlapping partitions on the datasets. The following Tables 8–11 and Figure 7 show the CN, EQ, $Q_{ov}$, and AC values of the results of AMI-MLPA algorithm on the datasets, compared with overlapping community detection algorithms Ego-splitting [44], DEMON [25], and BigClam [45]. Note that smaller values are better when using AC.

As illustrated above, the algorithm AMI-MLPA correctly figured out the number of communities on most of the networks. Although it does not get the best result on every dataset, but in most of the networks, the structural merits of the overlapping partitions found by AMI-MLPA are often better than the other algorithms.
6. Conclusion

In this paper, a multi-label propagation community detection algorithm based on average mutual information AMI-MLPA is proposed. The performance of the algorithm evaluated on real-world and synthetic datasets shows that it has a strong ability of community detection. In the experiment, we also observed that when finding overlapping communities on some datasets, although some algorithms get the wrong number of communities, the community structures detected seemed to be better, measured by EQ, Q_N, and AC. This may mean that for a particular network, the number of communities does not have to be a fixed value. In other words, different partitions can be obtained at different resolutions. In future research, we will focus on the research on the granularity of communities and the resolution of the community detection algorithms, to make our algorithm adaptively detect community structures of different granularities. At present, our primitive idea is to establish the granularity preference curve of the algorithm through parameter-based random networks and then modify the partitions obtained by the algorithm according to the curve for different granularity communities. We will also try to reduce the time complexity of the algorithm to make it suitable and efficient for larger-scale networks.

Data Availability

The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This research was funded by the Science and Technology Program of Guangzhou City (Grant No. 201707010052) and the Natural Science Foundation of Guangdong Province, China (Grant No. 2020A1515010696).

References

[1] J. M. Hofman, A. Sharma, and D. J. Watts, "Prediction and explanation in social systems," *Science*, vol. 355, no. 6324, pp. 486–488, 2017.
[2] S. Fortunato and C. Castellano, "Community structure in graphs," *Computational Complexity: Theory, Techniques, and Applications*, pp. 490–512, 2012.
[3] 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, p. 036106, 2007.
[4] Q. Dai, M. Guo, Y. Liu, X. Liu, and L. Chen, "MLPA: detecting overlapping communities by multi-label propagation approach," *IEEE Congress on Evolutionary Computation*, pp. 681–688, Cancun, Mexico, 2013.
[5] M. Girvan and M. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, 2002.
[6] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, p. 026113, 2004.
[7] Y. Zhang, B. Wu, and Y. Liu, "A novel community detection method based on rough set K-means," *Journal of Electronics and Information Technology*, vol. 39, no. 4, pp. 770–777, 2017.
[8] S. Agrawal and A. Patel, "SAG cluster: an unsupervised graph clustering based on collaborative similarity for community detection in complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 563, article 125459, 2021.
[9] Z. Wang, C. Wang, C. Gao, X. Li, and X. Li, "An evolutionary autoencoder for dynamic community detection," *SCIENCE CHINA Information Sciences*, vol. 63, no. 11, pp. 1–16, 2020.
[10] Y. Chen, C. Wang, and D. Li, "MINC-NRL: an information-based approach for community detection," *Algorithms*, vol. 15, no. 1, p. 20, 2022.
[11] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, vol. 11, no. 3, p. 033015, 2009.
[12] C. Lee, F. Reid, A. McDaid, and N. Hurley, "Detecting highly overlapping community structure by greedy clique expansion," 2010, http://arxiv.org/abs/1002.1827.
[13] Y. Li, R. Liu, Z. Qiu, and J. Zhu, "Community detection algorithm with locally social spider optimized," 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOECC), 2017, pp. 1178–1181, Chongqing, China, 2017.
[14] S. E. Garza and S. E. Schaeffer, "Community detection with the label propagation algorithm: a survey," *Physica A: Statistical Mechanics and its Applications*, vol. 534, p. 122058, 2019.
[15] S. Gregory, "Finding overlapping communities in networks by label propagation," *New Journal of Physics*, vol. 12, no. 10, p. 103018, 2010.
[16] J. Xie, B. K. Szymanski, and X. Liu, "SLPA: uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process," in *2011 IEEE 11th International Conference on Data Mining Workshops*, pp. 344–349, Vancouver, BC, Canada, 2011.
[17] Z. H. Wu, Y. F. Lin, S. Gregory, H. Y. Wan, and S. F. Tian, "Balanced multi-label propagation for overlapping community detection in social networks," *Journal of Computer Science and Technology*, vol. 27, no. 3, pp. 468–479, 2012.
[18] J. Ding, X. He, J. Yuan, Y. Chen, and B. Jiang, "Community detection by propagating the label of center," *Physica A: Statistical Mechanics and its Applications*, vol. 503, pp. 675–686, 2018.
[19] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.
[20] G. Yang, W. Zheng, C. Che, and W. Wang, "Graph-based label propagation algorithm for community detection," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 6, pp. 1319–1329, 2020.
[21] Y. Zhang, Y. Liu, R. Jin, J. Tao, L. Chen, and X. Wu, "GGLPA: a graph layout based label propagation algorithm for community detection," *Knowledge-Based Systems*, vol. 206, no. 9, p. 106363, 2020.
[22] Y. Su, B. Wang, F. Cheng, L. Zhang, X. Zhang, and L. Pan, “An algorithm based on positive and negative links for community detection in signed networks,” *Scientific Reports*, vol. 7, no. 1, pp. 1–12, 2017.

[23] P. Z. Li, L. Huang, C. D. Wang, and J. H. Lai, “Edmot: an edge enhancement approach for motif-aware community detection,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 479–487, Anchorage, AK, USA, 2019.

[24] Y. Zhang, B. Wu, Y. Liu, and J. Lv, “Local community detection based on network motifs,” *Tsinghua Science and Technology*, vol. 24, no. 6, pp. 716–727, 2019.

[25] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi, “Demon: a local-first discovery method for overlapping communities,” in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery & data mining*, pp. 615–623, Beijing, China, 2012.

[26] B. Guidi, A. Michienzi, and L. Ricci, “Sonic-man: A Distributed Protocol for Dynamic Community Detection and Management,” in *IFIP International Conference on Distributed Applications and Interoperable Systems*, pp. 93–109, Cham, 2018.

[27] E. A. Leicht, P. Holme, and M. E. J. Newman, “Vertex similarity in networks,” *Physical Review E*, vol. 73, no. 2, p. 026120, 2006.

[28] Z. Y. Zhao, H. Yu, Z. L. Zhu, and X. F. Wang, “Identifying influential spreaders based on network community structure,” *Chinese Journal of Computers*, vol. 37, pp. 753–766, 2014.

[29] S. P. Borgatti and M. G. Everett, “A graph-theoretic perspective on centrality,” *Social Networks*, vol. 28, no. 4, pp. 466–484, 2006.

[30] A. P. Kartun-Giles and G. Bianconi, “Beyond the clustering coefficient: a topological analysis of node neighbourhoods in complex networks,” *Chaos, Solitons & Fractals: X*, vol. 1, p. 100004, 2019.

[31] Z. M. Han, Y. Chen, W. Liu, B. H. Yuan, M. Q. Li, and D. G. Duan, “Research on node influence analysis in social networks,” *Journal of Software*, vol. 28, no. 1, pp. 84–104, 2017.

[32] D. B. Chen, H. Gao, L. Li, and T. Zhou, “Identifying influential nodes in large-scale directed networks: the role of clustering,” *PLoS One*, vol. 8, no. 10, article e77455, 2013.

[33] 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.

[34] T. Zhou, L. Lu, and Y. C. Zhang, “Predicting missing links via local information,” *The European Physical Journal B-Condensed Matter and Complex System*, vol. 71, no. 4, pp. 623–630, 2009.

[35] E. Nardelli, G. Proietti, and P. Widmayer, “Finding the most vital node of a shortest path,” *Theoretical Computer Science*, vol. 296, no. 1, pp. 167–177, 2003.

[36] D. Li, M. Q. Cheng, Y. Xu, F. Yuan, Y. N. Cheng, and Y. Q. Fu, “Optimal community detection method based on average mutual information,” *Science China*, vol. 59, no. 5, pp. 613–629, 2019.

[37] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*, Cambridge university press, 2003.

[38] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, “Comparing community structure identification,” *Journal of Statistical Mechanics*, vol. 2005, no. 9, p. P09008, 2005.