Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 18(2): 1609–1628.
DOI: 10.3934/mbe.2021083
Received: 03 November 2020
Accepted: 13 January 2021
Published: 02 February 2021

Research article

Parallel label propagation algorithm based on weight and random walk

Meili Tang[1], Qian Pan[1], Yurong Qian[2], Yuan Tian[3] , Najla Al-Nabhan[4] and Xin Wang [5,*]

[1] Nanjing University of information science Technology, Jiangsu, Nanjing 210044, China
[2] Xinjiang University, Urumqi 830008, China
[3] Nanjing Institute of Technology, Nanjing 211167, China
[4] Department of Computer Science, KingSaud University, Riyadh 11362, Saudi Arabia
[5] Huafeng Meteorological Media Group, Beijing 100080, China

* **Correspondence:** Email: xwang@cma.gov.cn.

Abstract: Community detection is a complex and meaningful process, which plays an important role in studying the characteristics of complex networks. In recent years, the discovery and analysis of community structures in complex networks has attracted the attention of many scholars, and many community discovery algorithms have been proposed. Many existing algorithms are only suitable for small-scale data, not for large-scale data, so it is necessary to establish a stable and efficient label propagation algorithm to deal with massive data and complex social networks. In this paper, we propose a novel label propagation algorithm, called WRWPLPA (Parallel Label Propagation Algorithm based on Weight and Random Walk). WRWPLPA proposes a new similarity calculation method combining weights and random walks. It uses weights and similarities to update labels in the process of label propagation, improving the accuracy and stability of community detection. First, weight is calculated by combining the neighborhood index and the position index, and the weight is used to distinguish the importance of the nodes in the network. Then, use random walk strategy to describe the similarity between nodes, and the label of nodes are updated by combining the weight and similarity. Finally, parallel propagation is comprehensively proposed to utilize label probability efficiently. Experiment results on artificial network datasets and real network datasets show that our algorithm has improved accuracy and stability compared with other label propagation algorithms.

Keywords: social network; community detection; label propagation; weight; random walk

1. Introduction

With the rapid development of the Internet, social networks are quickly entering people's life, which makes the idea of "all things interconnected" possible. The surge in the number of users has led to a

massive increase in online personal information, which has formed a variety of network structures from simple to complex. Personal information includes trajectories of their activities and connections with other individuals or groups, and the opinions and ideas they express are rapidly gaining popularity with the emergence of online social networks [1]. With the increasing popularity of sina weibo, WeChat, Facebook and Twitter, as a new product, social network has attracted the attention of many scholars in the field of data mining and analysis. Studying the communities in the network can help understand the structure and function of the entire network, analyze and predict the interactions between the elements of the entire network. Social networks, as a new type of communication model for people, usually contain a large amount of content data and link data that can be used for analysis [2, 3]. Content data may contain text, images, audio, and other multimedia data, and the nature of link data is the communication among entities. Therefore, this new communication mode provides unprecedented rich data for data mining research, from which huge benefits can be obtained through analysis and research [4, 5].

Community detection plays an important role in multiple areas such as e-commerce, precision advertising, and epidemic prevention and control [6]. With the development of community detection technology, great achievements have been made by studying data from various industries [7]. While the data changes people's life style, the huge amount of data brings great challenges, the network structure becomes more complex, and the research of stable and efficient algorithm becomes an urgent problem. At present, there are many community detection algorithms, but these algorithms are only suitable for small datasets. When dealing with massive data, there are some disadvantages, such as slow speed, instability, poor scalability and so on [8]. Therefore, it is necessary to conduct a deeper research on the network to cope with this change.

In this paper, we propose an algorithm based on the Label Propagation Algorithm (LPA). Firstly, the LPA algorithm automatically updates the label of the node until it converges. In the process of label selection, how to select the neighbor node and select which neighbor node to update the label of the current node will affect the partitioning of the label propagation algorithm. Therefore, weight is used to distinguish the influence of nodes. Generally speaking, the nodes with higher weights are more important in the network. Through the calculation of weights, the stability of the algorithm is greatly improved. Secondly, the method of parallelizing the algorithm to adapt for the large dataset is another point, and the parallel algorithm has good performance on the large data set.

Generally, the main contributions of our paper are as follows:

1). On the basis of label propagation, we propose Label Propagation Algorithm based on Weight and Random Walk (WRWLPA), which calculates the weights for each node, and combines the idea of random walks to calculate similarity.

2). After computing weights and similarities, we further parallelize the algorithm and propose Parallel Label Propagation Algorithm based on Weight and Random Walk (WRWPLPA).

3). We have done comparative experiments on artificial network datasets and real network datasets to prove the effectiveness of WRWPLPA algorithm, improves accuracy and stability compared to other label propagation algorithms.

The rest of the paper is organized as follows. Section 2 briefly presents the related works in the field of community detection, the original label propagation algorithm and its shortcomings are described. Section 3 gives a detailed introduction to the new improved parallel label propagation algorithm based on weights and random walk (WRWPLPA), followed by experiments and its results in Section 4.

Finally, the conclusion and future work are discussed in Section 5.

2. Related works

In recent years, community detection algorithms have become a hot topic in social networking, and have achieved very gratifying results in various fields, such as 5G networks [9]. Early community detection algorithms include graph partitioning algorithm [10], hierarchical clustering algorithm [11], partition clustering algorithm [12]. Nguyen et al. [13] improved the greedy algorithm K-L algorithm, the core idea is to use the greedy algorithm to maximize the cost function, so as to achieve the purpose of community partitioning.

The idea of hierarchical clustering method is to divide the whole network into a hierarchical community structure according to the similarity among nodes in the network, also known as the split method. The split method considers the whole network as a large community at the beginning, and gradually deletes the edge between the lowest similar nodes in the network, thus dividing the network into a small community until the terminating condition is reached. In addition, according to the specific scenarios and demands, we can choose the partition result at any time as the final partition result. In 2002, Newman et al. [14] propose the classical GN algorithm, which is the most typical top-down method. GN algorithm achieves the goal of community partition by removing the largest number of edges in the network. Since the time complexity of GN algorithm is high, it is only applicable to small scale networks. At the beginning of the condensation method, each node is regarded as a separate community, and a small community is merged by a certain custom standard.

In 2004, the concept of modularity was proposed to evaluate the quality of community detection [15]. The more obvious the community structure is, the higher the modularity is. After that, modularity has become a standard to measure the quality of community detection algorithm, and the problem of community detection has been transformed into the problem of modularity optimization. The algorithm FN [15] combines the communities with the highest modularity increment in each iteration, and divides the corresponding modules into the final result. Then there are many modularity optimization variants, such as greedy algorithm [15], simulated annealing algorithm [16], extreme optimization algorithm [17] and spectral optimization algorithm [18]. Steve Gregory et al. [19] proposed an algorithm named CONGA that uses node splitting to discover overlapping communities, and introduces the concept of local intermediation based on CONGA, proposes an improved algorithm called CONGO [20]. Although modularity is widely used, it still has certain limitations. For example, when modularity is maximized, small communities cannot be found [21]. In addition, extreme degradation of modularity also exists, which requires further exploration by researchers [22].

In 2005, Palla et al. [23] propose a CPM (Clique Percolation Method) algorithm based on the problem described by the overlapping community. CPM uses a connected complete subgraph (Clique) to discover the structure of the overlapping community. The time complexity of the algorithm is high because of the search for K-Clique in the network. On the basis of CPM, many scholars and researchers have made improvements, one of which is to extend the CPM algorithm to weighted networks and the bipartite graphs. In addition, Kumpula et al. [24] also proposed a fast factional filtering algorithm SCP, which is limited to the selection of initialization parameter K, and the community structure is different according to different K values.

In 2007, Raghavan et al. [25] proposed a classic label propagation algorithm. In each iteration, the

algorithm selects the label that appears most frequently in the neighbor nodes. The time complexity is linear, which can quickly discover the community, so it can be applied to large complex networks. In 2009, Leung et al. [26] introduce the weight value (score) for each label to measure the propagation capacity of the label. In 2011, Xie et al. [27] introduced the concept of speaker and listener on the label propagation algorithm. The core idea is to select a node as a listener in the process of updating the node, all of its neighbor nodes are its speakers. Multiple speakers send label information to the listener. Users need to customize the rules for listener to accept the label. At the same time, the algorithm saves the label of the node selected in each iteration, it calculates the frequency of each label in the history labels after stopping the iteration. The label with higher frequency is the label of this node. Tinghuai Ma et al. [28] propose a label propagation algorithm based on probability and similarity, using the propagation probability between nodes to discover the community and achieved good results.

COPRA [29] is proposed based on the label propagation algorithm in 2010. By specifying a global parameter k, each node is allowed to have k labels, that is, belonging to multiple communities. This is the first time to apply the label propagation algorithm to the overlapping community detection. In 2012, Zhihao Wu et al. [30] proposes a balanced multi-label propagation algorithm, which introduced the concept of the belonging coefficient under the conditions that each node allowed to belong to multiple communities. In the process of label propagation, each node gets the list of the label of its neighbor nodes, and accumulates the corresponding belonging coefficient, then normalizes and filters, and circulates the process until the stop condition is reached. Based on structural clustering, overlapping communities can also be found. Tinghuai Ma et al. [31–33] proposes an overlapping community detection algorithm LED (Loop Edges Delete), which has good performance in accuracy and efficiency. Imen et al. [34] introduced a Node Importance based Label Propagation Algorithm (NI-LPA) to detect overlapping communities in networks.

In recent years, many researchers try to discover community structure by different methods such as density-based and clustering, subgraph-based method [35,36]. Kai Lei et al. [37] propose a knowledge graph based solution for QEDL and developed a system consists of Question Entity Discovery (QED) module and Entity Linking (EL) module. Zijing Liu et al. [38] present a graph-theoretical approach to data clustering, which combines the creation of a graph from the data with Markov Stability, a multiscale community detection framework. Recently, Jia Li et al. [39] extend the adversarial graphs to more difficult community detection problems, focusing on black box attacks, and aim to hide targeted individuals from the detection of deep graph community detection models. Zhang et al. [40] proposed a Graph Layout based Label Propagation Algorithm to avoid the inaccurate and unstable in community detection as the node order of label updating and the mechanism of label propagation are random. And also, they used this algorithm as a Core Drug Discovery method in Traditional Chinese Medicine [41].

Researchers have made many attempts in community detection, incorporating advanced algorithms and techniques into community detection. Nevertheless, the current algorithm still has problems such as inability to process large-scale data, and lack of accuracy and stability. Different from the previous algorithms, this paper proposes a new similarity calculation method based on node weights and random walks in community detection, and uses the weights and similarities to update labels during the label propagation process. This makes the model proposed in this paper have better performance in accuracy and stability.

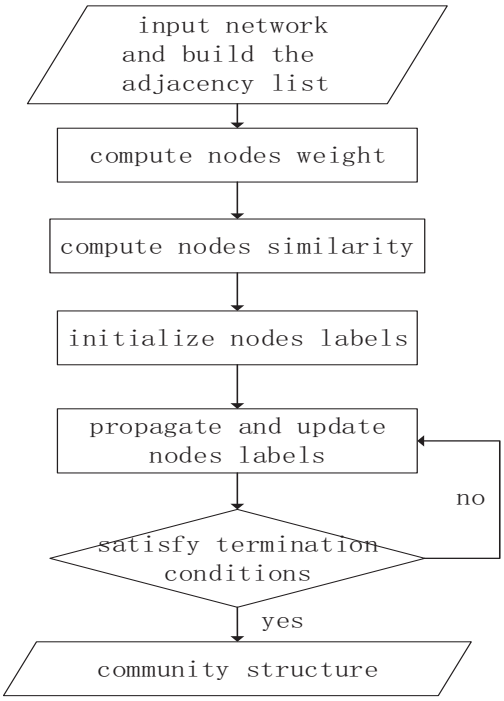

Figure 1. General procedures of WRWPLPA.

3. Parallel label propagation algorithm based on weight and random walk

3.1. Label propagation algorithm based on weight and random walk

In the traditional label propagation algorithms, every node updates its labels by considering the labels of its neighbors, there is no difference between neighbor nodes and they are all treated equally. In fact, even for the same node, different neighbor nodes may have different influence, so it is reasonable to assign a weight to each node. In our previous work [28], we proposed a parallel label propagation algorithm based on probability and similarity, which can effectively weight nodes in large-scale social networks. In this paper, we use the existed node weight calculation method [28] and propose a parallel label propagation algorithm based on weight and random walk (WRWPLPA). We select and update labels in the following steps. First, we use the method in our previous work to measure the weight of nodes. Where after, we use the concept of random walk to describe the similarity between each pair of nodes. Finally, weight and similarity are combined to update nodes' labels.

The general procedures of our algorithm are shown in Figure 1.

3.2. Parallel label propagation algorithm based on weight and random walk

In this paper, we use GraphX [42] for parallelization. GraphX is a component in Spark for graphs and graph-parallel computation. GraphX extends the Spark RDD as VertexRDD and EdgeRDD to support graph computation. While we calculating the weight of each node, the GraphX can provide Map-Reduce operators to parallel computing. The GraphX also provide bulk-synchronous parallel messaging mechanism to synchronize the super steps. What we realize the parallelization is use the

operators of Graphx.

The parallelization of WRWLPA (label propagation algorithm based on weight and random walk) contain four steps: graph construction and neighbor collection, node weighting, similarity calculation using random walk as well as node label initialization and selection.

3.2.1. Parallel graph construction and neighbor collection

Nowadays, social network datasets are so large that a single computer does not have enough memory to store information. Typically, it is stored in a distributed file system, such as the HDFS (Hadoop Distributed File System) or HBase (Hadoop database). Based on the parallel computation framework Spark, GraphX can load the network data from the distributed file system and establish the adjacency list with the given operator. There are many existing operators in GraphX: edgeListFile operator provides a way to load a graph from an edge list, such as a txt format file, where each line contains two node IDs; collectNeighbors and collectNeighborsIds operators can easily calculate the ID of the neighbor node and the label information it carries. When a graph is constructed, each node has an attribute with a default value of 1. In original graph, the tuple $(a, 1)$ means the node ID and its default attribute. The default attribute of each node can be changed by other operators, such as collectNeighborIds operator can change the default value to NID, which represents the ID of the node's neighbor.

3.2.2. Parallel nodes weighting

In our previous work, we proposed an efficient method to measure weight of nodes. Nodes located in the core of the network are the most important nodes. These nodes can be discovered by the k-shell decomposition method. However, we can not use only K-shell method to calculate the weight of nodes.

Degree centrality is a typical type of local metric, which is very simple but ignores the information carried by the node itself. For example, if the importance of a node is measured only by its degree, all leaf nodes in the network will have the same importance. In fact, the importance of different leaf nodes is different. At the same time, the location of nodes in the network also affects its importance. In general, the nodes in the core of the network tend to be more influential. In this paper, the neighborhood index and position index are calculated by combining the neighborhood attribute and location attribute of nodes. We combine the two indexes to get the final weight of nodes.

Definition 1. Given a complex social network $G(V, E)$, for any node $n_i \in V$, the capability of node n_i to influence the neighborhood attribute is denoted as $ICN(n_i)$, is given by:

$$ICN(n_i) = \frac{d(n_i) + \sum_{n_j \in N(n_i)} d(n_j)}{\max\limits_{n_i \in V} \left(d(n_i) + \sum_{n_j \in N(n_i)} d(n_j)\right)} \tag{1}$$

where, the degree of node n_i and n_j are $d(n_i)$, $d(n_j)$, respectively. $N(n_i)$ represents the neighbor of n_i, $ICN(n_i)$ denotes the neighborhood index and $ICN(n_i) \in [0, 1]$. ICN can effectively distinguish the nodes with the same degree. The higher the ICN value, the more important the node is.

Definition 2. Given a complex social network $G(V, E)$, each node is given K_s value through the k-shell decomposition method. For the node $n_i \in G$, the K_s value of n_i is assumed to be k. For each round of K-shell decomposition, it is assumed that the number of iterations is m, and node n_i is removed in

the nth iteration of the k-degree process, $1 \leq n \leq m$. $W(n_i)$ denotes the weight of node n_i, defined as follows:

$$W(n_i) = K_s * e^{ICN(n_i)+n/m} \tag{2}$$

In Eq (2), n/m denotes the position index, the neighborhood index and position index are combined to calculate the weight of nodes. K_s is K-shell value, it means the node important, and the $ICN(n_i)$ indicate the important around the neighbor, and the n/m means the centrality of n_i, we use exponential form to enlarge the weight value. For every K_s , the weight vary from K_s to $K_s * e^2$.

3.2.3. Calculation of similarity using random walk

The random walk model is proposed as random move [28], is an irregular diffusion process. In the real world, many scenes can be expressed as random walk models, such as trajectories of ants walking, brown movement of pollen, node-to-node propagation paths in social networks and so on. Although these scenes are not completely random, the transition from the current state to the next state is random.

Due to the idea of label propagation, a node can propagate its own label to another node in a social network. Random walk is a Markov model [43], which can also be called a special case of the Markov chain. In a social network, a series of sequences accessed by a node can be obtained by random walks. The mathematical expression of random walk process is as shown in Eq (3):

$$X_t = X_{t-1} + e_t \tag{3}$$

Where X_t represents the value at time t, X_{t-1} represents the value at time $t-1$, and e_t represents the error between these two moments.

In this section, a random walk strategy will be used to describe the similarity between nodes. Except using random walks to calculate similarity, there are other different measures, which are the ACT (average commute time) [44] and MFPT (mean first passage time) [45]. These two methods are easy to understand, but they are not suitable for large-scale network computing due to their high complexity.

The core idea of random walk based label propagation algorithm (RWLPA) [46] is to use random walk to measure the distance between nodes. The farther apart the two nodes are, the less likely they are to belong to the same community. According to the method described in the RWLPA, the algorithm initially places a randomly walking walker on any node in the network and randomly selects the next walk position according to the Markov model. For convenience of description, P_{xy} is used to represent the probability that a random walker walks from node x to y in one step, and $\pi_{xy}(t)$ is used to represent the probability that random walker walks from node x to node y through t steps, and uses $\pi_x(t)$ represents the row vector of the x-th row of the matrix $\pi(t)$. P_{xy} and $\pi_x(t)$ are calculated as follows:

$$P_{xy} = \frac{a_{xy}}{d(x)} \tag{4}$$

$$\pi_x(t) = \pi_x(t-1) \cdot P \tag{5}$$

Where P_{xy} is an element of the x-th row and y-column of the probability transfer matrix P. If there is an edge between node x and node y in the network, the value of a_{xy} is 1; if there is no edge connection, the value of a_{xy} is 0. The value of $d(x)$ represents the degree of node x. P_{xy} means that Walker will

randomly select the next node according to the degree of the node x. If there is no edge connected between node x and node y, the value of P_{xy} is 0. In other words, walker in the random walk cannot get from node x to node y directory. But after t step iteration, P_{xy} maybe is not 0. Use Sim_{xy} to represent the similarity between node x and node y. The mathematical calculations are as follows:

$$Sim_{xy}(t) = d(x) \cdot \pi_{xy}(t) + d(y) \cdot \pi_{yx}(t) \tag{6}$$

In the above expression, the value of $d(x)$ represents the degree of node x. We use this form to indicate the profanities from node x transfer to node y and from node y to node x, means whatever node x and node y has connectivity, they have similarity. Assuming that node x and node y belong to the same community, the theoretical similarity should be very high. However, due to the random nature of random walks, a walker may walk to other communities or nodes that are far away, resulting in a low degree of similarity between node x and node y. This is contrary to the fact and cannot achieve the expected effect. In order to solve this problem, we can use a random walk strategy to accumulate the similarity of the asynchronous numbers, and then calculate the average value to reduce the uncertainty of a random walk. The mathematical formula for accumulating the averaging is as follows:

$$Sum\_AvgSim_{xy} = \frac{\sum_{i=1}^{t} Sim_{xy}(t)}{t} \tag{7}$$

The importance of all nodes in RWLPA is the same, ignoring some inherent information of nodes in the network, which leads to the inaccuracy of the final community division result. To solve this problem, nodes must be assigned weights to distinguish between important and unimportant nodes in the network. We combine the weights of nodes based on the similarity of random walk calculations, so that the similarity between nodes can be calculated more accurately. Assuming that the weights of node x and node y in the network are calculated as $W(x)$ and $W(y)$ by the Eq (2). We combine the weighted and similarity together, to calculated the similarity of node y according to node x as follows:

$$NewSim_{(x,y)} = \frac{W(y)}{W(x) + W(y)} + Sum\_AvgSim_{xy} \tag{8}$$

Based on the above calculation of node weight and similarity between nodes, Label Propagation Algorithm based on Weight and Random Walk (WRWLPA) is proposed. The pseudo-code of the algorithm is shown in algorithm 1.

3.2.4. Parallel Label Propagation Algorithm based on Weight and Random Walk

Combine with the above steps, we propose Parallel Label Propagation Algorithm based on Weight and Random Walk (WRWPLPA). Firstly, the probability transfer matrix between nodes is calculated through random walk. As the number of steps t increases, the probability transfer matrix will tend to a stable state. If the probability transfer matrix at this time is used to calculate the similarity between nodes, the similarity does not depend on other parameters, but only on the degree of nodes. Therefore, a higher value of t does not mean that the obtained similarity matrix is more accurate. The empirical value $t = 5$ is used here [28]. Assume that the probability transfer matrix at time $t = 0$ is A, then the probability transfer matrix at time $t = 1$ is $AA = A^2$, and so on. Finally, the probability transfer matrix $A_i(i = 0, 1, 2, 3, 4, 5, \cdots)$ at each moment is calculated, then the similarity matrix S is obtained according to Eq (7).

Algorithm 1: Pseudo-code of WRWLPA

Input: $G = (V, E)$.
Output: community structure of the network.

```
load data;
for n in V do
    give the node a unique ID value;
end
calculate the weight of each node according to equation(2);
calculate the similarity of each node according to equation(8);
while 1 do
    for n in V do
        find the most frequent labels among the neighbours of node n;
        if multiple labels have appeared most times then
            find the node that maximizes NewSim and update the labels of the node n based on
            maximizes NewSim node;
            if multiple nodes with the highest similarity then
                find the node with the highest weight among neighbors;
                update label according to the node with maximum weight;

            else
                update label according to the node with maximum similarity;
            end
        else
            update label according to the node with the most frequent labels;
        end
    end
    flag = true;
    for n in V do
        if the label of the node is inconsistent with the most frequent labels in the neighbor nodes
        then
            set flag=false;
        end
    end
    if flag=true then
        reach stop condition and then end;
    end
end
```

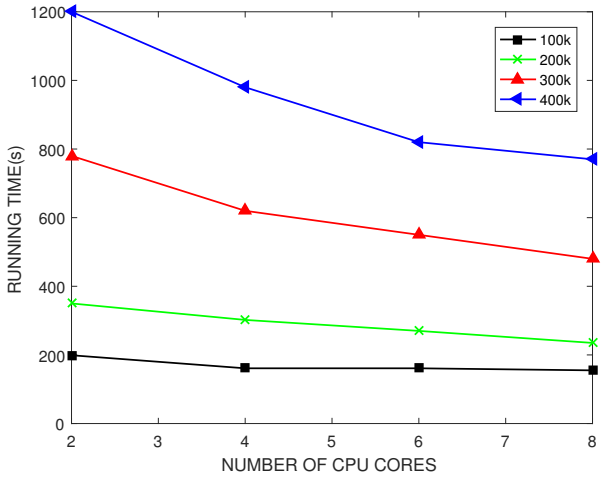

Figure 2. After incorporating the weights, the running time of the model on different cores.

Then, using Spark's broadcast mechanism, the similarity matrix S is broadcast to the entire cluster. According to Eq (8), each node can calculate the updated similarity matrix S_new by combining the weights. After the weight and similarity matrix have been calculated, the label can be propagated.

In the label initialization stage, two-tuple $(label_i, p)$ is used to replace the default attribute value of node, where $label_i$ represents the label of node, generally its node ID, and p is the probability that the node belongs to this label. Since each node is a separate community, it is initialized using a two-tuple $(NID, 1.0)$, where NID is the ID of the node. In the label propagation phase and label selection phase, the Pregel operator is used to send the label list information to its neighbor nodes. Three custom functions need to be implemented in the Pregel operator. The sendMessage function and the mergeMessage function are used to send messages and merge messages respectively; the VertexProgram function is used to implement a custom node update strategy.

All the steps above are combined to achieve the parallelization of the algorithm WRWLPA. The pseudo-code of the algorithm is shown in Algorithm 2.

3.3. Time complexity analysis

WRWPLPA can significantly improve the accuracy and stability of the label propagation algorithm by adding the weights of nodes and incorporating the concept of random walk. This section analyzes its time complexity. Assume that the network has m edges, n nodes and the average degree of the nodes is k. As we all know, the label propagation algorithm can be performed using a linear time. The WRWPLPA proposed in this section is not as good as the label propagation algorithm in time efficiency, but it performs better in accuracy and stability.

For the initialization of each node as a separate community, the time complexity is $O(n)$. In the calculation of node weights, the K-Shell method is used, and the time complexity is also linear. As shown in Figure 2, as the number of CPUs increases, the runtime of the algorithm decreases.

However, when using random walk to calculate the similarity matrix, the time complexity is $O(n^3)$, which greatly increases the time cost of our method. Because our method is mainly dedicated to

Algorithm 2: Pseudo-code of WRWPLPA

Input: $G = (V, E)$.
Output: community structure of the network.

```
// Graph Construction and Neighbor Node Information Collection;
constructing a graph using the edgeListFile operator;
collect node degree information using outerJoinVertices operator;
collect neighbor node ID information using the collectNeighborIds operator;
// Calculate node weights;
k = 1, iter = 1;
while graph is not empty do
    find nodes with degree of k;
    while graph is not empty do
        add the found node and iter values to the collection;
        iter = iter + 1;
        calculate sub-graph after removing nodes;
        get the node set of degree k;
    end
    k = k + 1;
end
calculate node weights according to equation(2);
for nodes in graph do
    get similarity matrix Sim from broadcast variables; according to formula (6)
    receive weights, calculate updated similarity matrix NewSim; according to formula (8);
end
//Node initialization;
for nodes in graph do
    the initialization label is the node's own ID;
end
//Label propagation and selection;
while not reach the maximum number of iterations do
    for nodes in graph do
        Send label list;
    end
    for nodes in graph do
        update the label according to the logic of label update in Algorithm 1;
    end
end
Output the final community;
```

improving accuracy and stability, based on comprehensive considerations, the time cost is not described too much. The time complexity of each label propagation is $O(m)$. In summary, the time complexity of WRWPLPA is $O(n^3 + m \cdot iter)$ ($iter$ is the number of iterations of the algorithm).

4. Experimental design and results analysis

There are two ways to measure the quality of community division. For the real network datasets, it is usually unable to accurately measure the community to which each entity belongs, so the modularity is used to measure the accuracy of community division. The higher the modularity is, the better the community division is. For the artificial network datasets, various parameters can be specified, such as the number of specified partitions, the size of partitions and so on. After the algorithm divides the community, the difference between the division results and the preset values can be evaluated by calculating the normalized mutual information value between the two results. The algorithms in this section will be validated and evaluated on the artificial network datasets and the real network datasets. All algorithms run on a 2.6 GHz i7 processor, 16 GB computer.

4.1. Real network dataset

There are many real network datasets that have been abstracted into the realm of community division. For example, the karate club network [47], which is a classic network dataset, is small and representative. This network dataset has 34 nodes and 78 edges, the nodes represent the members of the club, and the edges represent the relationship between members and members. Figure 3 shows the network structure of the karate club. As shown in Figure 3, the entire network is divided into two communities. Among them, node 1 represents the coach, and node 34 represents the club manager. In addition to karate clubs, we also selected two network datasets, namely Polbooks Network dataset [48] and Dolphin Network dataset [49]. The Polbooks dataset contains 105 nodes and 441 edges. The nodes of the network represent books related to US politics sold on the Amazon Online Bookstore, and the edges of the network represent the number of readers who bought both books at the same time. For the dolphin network dataset, it is a network of 62 nodes with 159 edges.

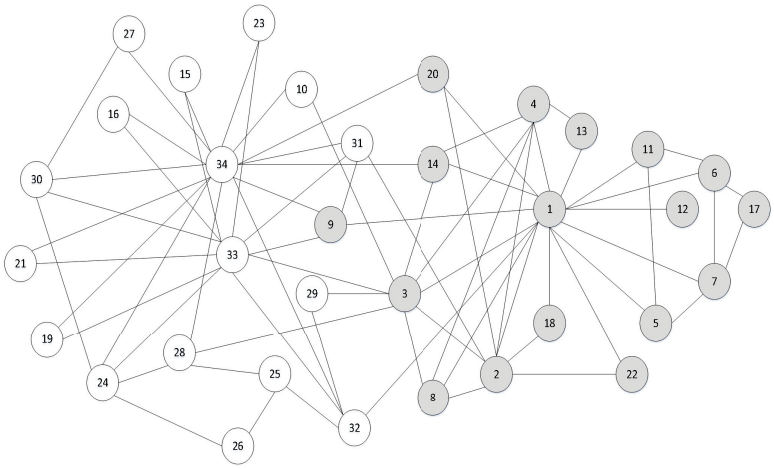

Figure 3. Karate club network.

In this section, the three real network datasets will be used to evaluate the algorithms. The algorithms used are the WRWPLPA proposed in this section, as well as the GN algorithm, the FN algorithm, the label propagation algorithm LPA and the RWLPA mentioned in related works section. In addition, we also compared with LPALC (Label Propagation Algorithm based on Local Cycles) [50] and NILPA (New Improved Label Propagation Algorithm) [51]. The LPALC algorithm improves the random selection process that exists in the LPA algorithm. When random selection is required, the label of the neighbor node with the smallest node ring is selected. NILPA improves model performance by incorporating weights in label propagation algorithms. We use modularity to measure the performance of the algorithm on different datasets. Table 1 shows the comparison of experimental results for different algorithms on different datasets.

Table 1. The modularity obtained by each algorithm on three datasets.

Datasets	$Q(GN)$	$Q(FN)$	$Q(LPA)$	$Q(LPALC)$	$Q(RWLPA)$	$Q(NILPA)$	$Q(WRWPLPA)$
Karat Club	0.3581	0.3461	0.3894	0.4235	0.3916	0.3715	0.3918
Dolphin	0.5104	0.4803	0.4845	0.5037	0.5037	0.5265	0.5286
PolBooks	0.5168	0.5020	0.4986	0.4069	0.5081	0.5209	0.5274

From Table 1, it can be seen that the WRWPLPA algorithm proposed in this section has achieved higher modularity on most case except Karat Club datasets. Karat Club dataset has obvious two communities and the network topology is simple. It means for simple, small network, our algorithm has no advantages. The experimental results show that the WRWPLPA algorithm is superior to the four baselines on real network datasets in most cases.

4.2. Artificial network dataset

In this section, we use artificial datasets to verify the algorithm. The dataset uses the LFR benchmark network [52]. We can set the parameter to generate the networks which meet our demand, where n is the number of nodes, m is the number of edges, mu is the mixing parameters, k is the average degree of nodes, $maxk$ is the maximum degree of nodes, on is the number of overlapping nodes, om is the number of memberships of the overlapping nodes, $minc$ is the minimum community size and $maxc$ is the maximum community size. In particular, a network with a high value of mu means the community is difficult to be uncovered correctly. The mixing coefficient is set to 0.1 to 0.8. At the same time, the network size is set to 500, 1000, 1500, and 2000. The specific parameters are shown in Table 2. NMI (Normalized Mutual Information) is an important measure of community detection. NMI is often used to measure the similarity of two clustering results, which can objectively evaluate the accuracy of a community division compared with the standard division. The range of NMI is 0 to 1. The higher the NMI value, the more accurate the division.

Defining a confusion matrix N, where the rows stand for the real communities and the columns stand for the found communities which had been found. The member of N, N_{ij} is simply the number of nodes in the real community i that appear in the found community j. The number of real communities is denoted c_x and the number of found communities is denoted c_y, the sum over row i of matrix N_{ij} is denoted N_i and the sum over column j is denoted N_j. NMI is defined as:

The calculation formula of NMI is as follows:

Table 2. main parameter in LFR network

Datasets	Parameter
LRF Artificial network	Dataset scale n = 500,1000,1500,2000, mu = 0.1 to 0.8, k = 20, $maxk$ = 100, $minc$ = 50, $maxc$ = 100, $on = 0.1n$, om = 3

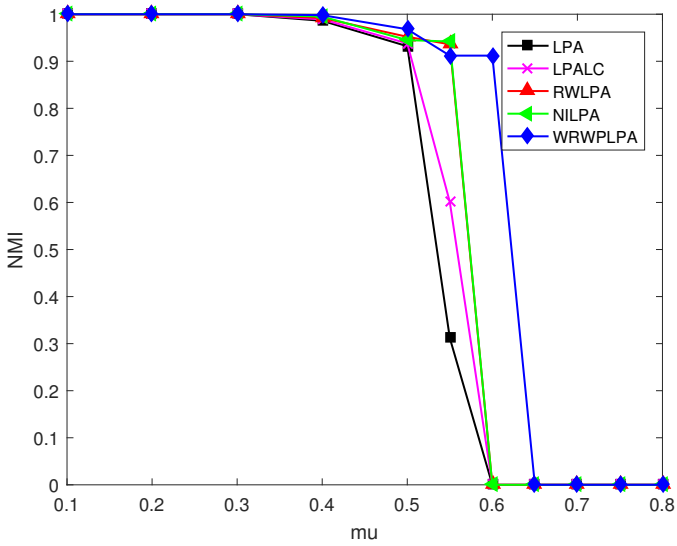

Figure 4. Result of 500-node network with different *mu*.

$$NMI(X,Y) = \frac{-2\sum_{i=1}^{c_x}\sum_{i=1}^{c_y} N_{ij}\log\left(\frac{N_{ij}N}{N_iN_j}\right)}{\sum_{i=1}^{c_y} N_j\log\left(\frac{N_j}{N}\right) + \sum_{i=1}^{c_x} N_i\log\left(\frac{N_i}{N}\right)} \tag{9}$$

In the experiment, we set the mixing coefficient to vary from 0.1 to 0.8. When *mu* is from 0.1 to 0.5, the change range is 0.1; when *mu* is from 0.5 to 0.8, the change range is 0.05. In other words, for each scale, there will be 11 networks with different mixing coefficients. For each network, use LPA, LPALC, RWLPA, NILPA, WRWPLPA for analysis and evaluation of experiments.

Figure 4 shows the results of an experiment on a 500-node network. It can be seen from the figure that when the mixing coefficient *mu* is less than 0.4, all three algorithms can obtain higher values. When *mu* is small, the community structure is obvious, so the algorithm can usually get better community division. When *mu* value is greater than 0.4, the value of LPA begins to decline. When *mu* is equal to 0.6, four baselines are no longer able to identify the communities, but WRWPLPA can also obtain a good detection of community structure. When the value is greater than 0.65, all the algorithms cannot find the community structure because the network becomes more complicated.

Figure 5 shows the network of 1000 nodes. It can be seen from the figure that all algorithms have good accuracy when *mu* is less than 0.4; when *mu* is greater than 0.55, the value obtained by LPA and LPALC decreases rapidly, which means that the algorithm is difficult to find the community structure;

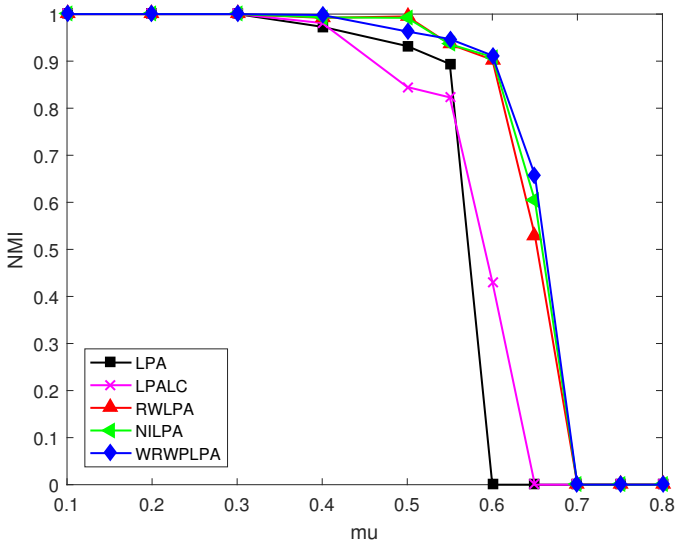

Figure 5. Result of 1000-node network with different *mu*.

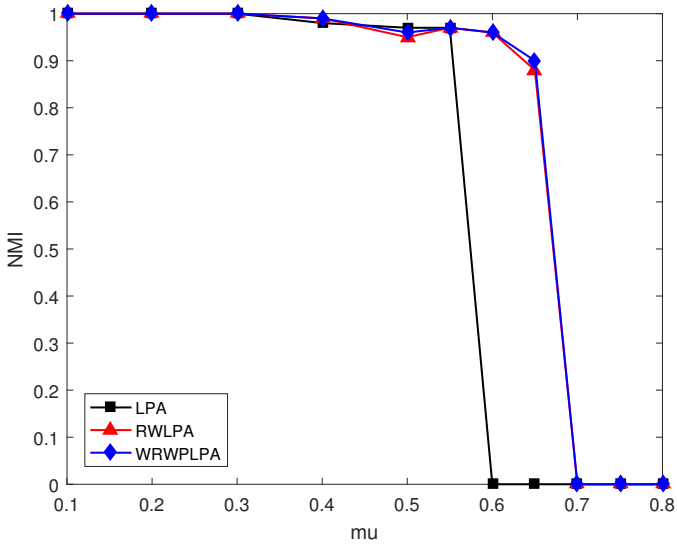

Figure 6. Result of 1500-node network with different *mu*.

when *mu* is greater than 0.6, RWLPA, NILPA, and WRWPLPA have a certain degree of decline, but the NMI value obtained by RWLPA and NILPA falls faster.

Figures 6 and 7 show the network of 1500 and 2000 nodes, respectively. The comparison of the values obtained by the various algorithms can be summarized as follows: When *mu* is less than 0.4, the algorithm can get better community division; when *mu* is more than 0.4, the quality of community classification obtained by the algorithm shows a downward trend, but the WRWPLPA declines more slowly, which fully demonstrates the accuracy and stability.

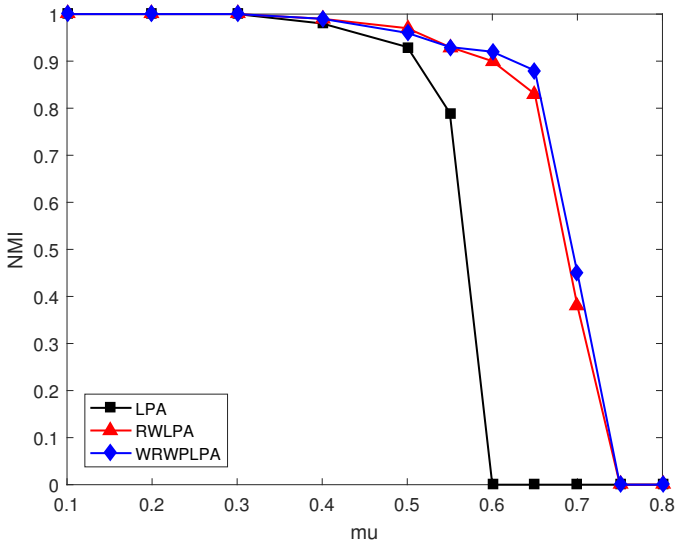

Figure 7. Result of 2000-node network with different *mu*.

5. Conclusions

The existing algorithms have the disadvantages of slow speed, instability and poor scalability when dealing with massive data. In this work, we present a Parallel Label Propagation Algorithm based on Weight and Random Walk (WRWPLPA). On the basis of calculating the weight of nodes, the idea of random walk is fused, and WRWLPA is proposed by combining the idea of label propagation, which has excellent performance in precision and accuracy.

Although the algorithm proposed in this paper have good performance in accuracy and stability, there is still space for improvement, such as WRWLPA does not optimize the label propagation algorithm thoroughly and cannot detect the dynamic network structure. Although the proposed algorithm has been parallelized, in the actual cluster, there are still many places that can be optimized. Future research can be conducted from the following aspects:

- It is hoped that in the future work, the characteristics of dynamic networks can be studied in depth, the incremental similarity computation can be designed to adapt to dynamic networks, and improve the practicability of the label propagation algorithm.
- We can study the optimization of the algorithm in parallelization, such as how nodes and edges are partitioned to improve efficiency of the algorithm, and how to dynamically add nodes in the network.

Acknowledgements

This work was supported in part by National Key Research and Development Program of China (2018YFC1507805). The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through research group no. RGP-1441-33.

Conflict of interest

The author declares no conflict of interest.

References

1. T. Ma, Y. Zhao, H. Zhou, Y. Tian, A. A. Dhelaan, M. Al-Rodhaan, Natural disaster topic extraction in sina microblogging based on graph analysis, *Expert Syst. Appl.*, **115** (2019), 346–355.
2. H. Rong, T. Ma, J. Cao, Y. Tian, A. A. Dhelaan, M. Al-Rodhaan, Deep rolling: A novel emotion prediction model for a multi-participant communication context, *Inf. Sci.*, **488** (2019), 158–180.
3. J. Wu, Z. Chen, M. Zhao, An efficient data packet iteration and transmission algorithm in opportunistic social networks, *J. Ambient Intell. Humanized Comput.*, **11** (2020), 3141–3153.
4. T. Ma, H. Rong, Y. Hao, J. Cao, Y. Tian, M. Al-Rodhaan, A novel sentiment polarity detection framework for chinese, *IEEE Trans. Affective Comput.*, **2019** (2019).
5. H. T. Nguyen, A. Cano, T. Vu, T. N. Dinh, Blocking self-avoiding walks stops cyber-epidemics: a scalable gpu-based approach, *IEEE Trans. Knowl. Data Eng.*, **32** (2020), 1263–1275.
6. C. Su, Y. K. Wang, Y. Yu, *Community detection in social networks*, Trans Tech Publications Ltd, 2014.
7. R. Kishore, S. Swayamjyoti, S. Das, A. K. Gogineni, Z. Nussinov, D. Solenov, et al., Visual machine learning: Insight through eigenvectors, chladni patterns and community detection in 2d particulate structures, *arXiv preprint arXiv:2001.00345*, 2020.
8. K. Liu, Z. Chen, J. Wu, Y. Xiao, H. Zhang, Predict and forward: An efficient routing-delivery scheme based on node profile in opportunistic networks, *Future Int.*, **10** (2018), 74.
9. M. Alsenwi, K. Kim, C. S. Hong, Radio resource allocation in 5g new radio: A neural networks based approach, *arXiv preprint arXiv:1911.05294*, 2019.
10. Y. Kang, B. Yu, W. Wang, D. Meng, *Spectral clustering for large-scale social networks via a pre-coarsening sampling based nystrm method*, Pacific-asia Conference on Knowledge Discovery & Data Mining, 2015.
11. D. Cheng, Q. Zhu, Q. Wu, *A local cores-based hierarchical clustering algorithm for data sets with complex structures*, 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), 2018.
12. J. Wu, L. Zhang, Y. Li, Y. Jiao, Partition signed social networks via clustering dynamics, *Phys. A*, **443** (2016), 568–582.
13. T. P. Q. Nguyen, R. Kuo, Partition-and-merge based fuzzy genetic clustering algorithm for categorical data, *Appl. Soft Comput.*, **75** (2019), 254–264.
14. M. E. J. Newman, Finding community structure in networks using the eigenvectors of matrices, *Phys. Rev. E*, **74** (2006), 036104.
15. M. E. Newman, Fast algorithm for detecting community structure in networks, *Phys. Rev. E*, **69** (2004), 066133.

16. R. Guimera, M. Sales-Pardo, L. A. N. Amaral, Modularity from fluctuations in random graphs and complex networks, *Phys. Rev. E*, **70** (2004), 025101.

17. J. Duch, A. Alex, Community detection in complex networks using extremal optimization, *Phys. Rev. E*, **72** (2005), 027104.

18. L. Donetti, M. A. Munoz, Detecting network communities: a new systematic and efficient algorithm, *J. Stat. Mech. Theory Exp.*, **2004** (2004), P10012.

19. S. Gregory, *An algorithm to find overlapping community structure in networks*, in European Conference on Principles & Practice of Knowledge Discovery in Databases, 2007.

20. S. Gregory, *A fast algorithm to find overlapping communities in networks*, Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2008.

21. P. Schuetz, A. Caflisch, Multistep greedy algorithm identifies community structure in real-world and computer-generated networks, *Phys. Rev. E*, **78** (2008), 026112.

22. F. Hu, J. Liu, L. Li, J. Liang, Community detection in complex networks using node2vec with spectral clustering, *Phys. A*, **545** (2020), 123633.

23. G. Palla, I. Derényi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, *Nature*, **435** (2005), 814–818.

24. J. M. Kumpula, K. Mikko, K. Kimmo, S. K. Jari, Sequential algorithm for fast clique percolation, *Phys. Rev. E*, **78** (2008), 026109.

25. R. Usha Nandini, A. Rka, K. Soundar, Near linear time algorithm to detect community structures in large-scale networks, *Phys. Rev. E*, **76** (2007), 036106.

26. I. X. Y. Leung, H. Pan, L. Pietro, C. Jon, Towards real-time community detection in large networks, *Phys. Rev. E*, **79** (2007), 066107.

27. J. Xie, S. Kelley, B. K. Szymanski, Overlapping community detection in networks: the state of the art and comparative study, *Acm Comput. Surv.*, **45**, (2013), 43.

28. T. Ma, W. Shao, Y. Hao, J. Cao, Graph classification based on graph set reconstruction and graph kernel feature reduction, *Neurocomputing*, **296** (2018), 33–45.

29. S. Gregory, Finding overlapping communities in networks by label propagation, *New J. Phys.*, **12** (2010), 103018.

30. Z. H. Wu, Y. F. Lin, S. Gregory, H. Y. Wan, S. F. Tian, Balanced multi-label propagation for overlapping community detection in social networks, *J. Comput. Sci. Technol.*, **27** (2012), 468–479.

31. T. Ma, Y. Wang, M. Tang, C. Jie, T. Yuan, A. Al-Dhelaan, M. Al-Rodhaan, Led: A fast overlapping communities detection algorithm based on structural clustering, *Neurocomputing*, **207** (2016), 488–500.

32. T. Ma, Q. Liu, J. Cao, Y. Tian, A. Al-Dhelaan, MznahAl-Rodhaan, Lgiem: Global and local node influence based community detection, *Future Gener. Comput. Syst.*, **105** (2020), 533–546.

33. T. Ma, Q. Pan, H. Wang, W. Shao, Y. Tian, N. Al-Nabhan, Graph classification algorithm based on graph structure embedding, *Expert Syst. Appl.*, **161** (2020), 113715.

34. I. Ben El Kouni, W. Karoui, L. B. Romdhane, Node importance based label propagation algorithm for overlapping community detection in networks, *Expert Syst. Appl.*, **162** (2020), 113020.
35. Y. Lv, T. Ma, M. Tang, J. Cao, Y. Tian, A. Al-Dhelaan, MznahAl-Rodhaan, An efficient and scalable density-based clustering algorithm for datasets with complex structures, *Neurocomputing*, **171** (2016), 9–22.
36. T. Ma, H. Wang, L. Zhang, Y. Tian, N. Al-Nabhan, Graph classification based on structural features of significant nodes and spatial convolutional neural networks, *Neurocomputing*, **423** (2021), 639–650.
37. K. Lei, B. Zhang, Y. Liu, Y. Deng, D. Zhang, Y. Shen, *A knowledge graph based solution for entity discovery and linking in open-domain questions*, International Conference on Smart Computing and Communication, 2017.
38. Z. Liu, M. Barahona, Graph-based data clustering via multiscale community detection, *Appl. Network Sci.*, **5** (2020), 1–20.
39. J. Li, H. Zhang, Z. Han, Y. Rong, H. Cheng, J. Huang, *Adversarial attack on community detection by hiding individuals*, WWW '20: Proceedings of The Web Conference, 2020.
40. Y. Zhang, Y. Liu, R. Jin, J. Tao, L. Chen, X. Wu, Gllpa: A graph layout based label propagation algorithm for community detection, *Knowl. Based Syst.*, **206** (2020), 106363.
41. Y. Zhang, Y. Liu, Q. Li, R. Jin, C. Wen, Lilpa: A label importance based label propagation algorithm for community detection with application to core drug discovery, *Neurocomputing*, **413** (2020), 107–133.
42. Graphx programming guide, 2020. Available from: http://spark.apache.org/docs/latest/graphx-programming-guide.html#graph-algorithms.
43. A. Bandyopadhyay, O. Zeitouni, Random walk in dynamic markovian random environment, *arXiv preprint math/0509066*, 2005.
44. W. Liu, L. Lü, Link prediction based on local random walk, *EPL*, **89** (2010), 58007.
45. P. Hanggi, P. Talkner, First-passage time problems for non-markovian processes, *Phys. Rev. A*, **32** (1985), 1934–1937.
46. X.-K. Zhang, C. Song, J. Jia, Z.-L. Lu, Q. Zhang, An improved label propagation algorithm based on the similarity matrix using random walk, *Int. J. Mod. Phys. B*, **30** (2016), 1650093.
47. W. W. Zachary, An information flow model for conflict and fission in small groups, *J. Anthropological Res.*, **33** (1977), 452–473.
48. W. Li, C. Huang, M. Wang, X. Chen, Stepping community detection algorithm based on label propagation and similarity, *Phys. A*, **472** (2017), 145–155.
49. D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, S. M. Dawson, The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations, *Behav. Ecol. Sociobiology*, **54** (2003), 396–405.
50. X. K. Zhang, S. Fei, C. Song, X. Tian, Y. Y. Ao, Label propagation algorithm based on local cycles for community detection, *Int. J. Mod. Phys. B*, **29** (2015), 1550029, 2015.

51. T. Ma, Z. Xia, An improved label propagation algorithm based on node importance and random walk for community detection, *Mod. Phys. Lett. B*, **31** (2017), 1750162.

52. A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, *Phys. Rev. E*, **78** (2008), 046110.

AIMS Press