World Wide Web
https://doi.org/10.1007/s11280-017-0519-0

A novel parallel community detection scheme based on label propagation

Naiyue Chen[1] · **Yun Liu**[1] · **Junjun Cheng**[2] · **Qing Liu**[3]

Received: 18 January 2017 / Revised: 3 November 2017 / Accepted: 6 December 2017

Abstract Community detection is one of the most important ways to reflect the structures and mechanisms of a social network. The overlapping communities are more in line with the reality of the social networks. In society, the phenomenon of some members sharing memberships among different communities reflects as overlapping communities in the networks. Dealing with big data networks, it is a challenging and computationally complex problem to detect overlapping communities. In this paper, we propose highly scalable variants of a community-detection algorithm in a parallel manner called Label Propagation with nodes Confidence (PLPAC). We introduce MapReduce into our scheme to process the big data in a parallel manner and guarantee the efficiency of community detection. We implemented the algorithm on artificial networks as well as real networks to evaluate the accuracy and speedup of the proposed method. Experimental results on datasets from different scenarios illustrate that the improved label propagation method outperforms the state-of-the-art methods in terms of accuracy and time efficiency.

Keywords Social network · Community detection · Label propagation · Parallel computation

✉ Yun Liu
liuyun@bjtu.edu.cn

Naiyue Chen
13111007@bjtu.edu.cn

Junjun Cheng
chengjj@itsec.gov.cn

Qing Liu
liuqing@powerchina.cn

[1] Key Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education, Beijing Jiaotong University, Beijing, China

[2] China Information Technology Security Evaluation Center, Beijing, China

[3] China Electric Power Construction Limited by Share Ltd, Beijing, China

Published online: 16 December 2017

1 Introduction

Many systems in the real word can be considered networks, such as disease transmission networks, DBLP data resources, protein–protein interaction networks, scientist collaboration networks, and social networks. With the rapid development of the networks, social networks have become an indispensable part of contemporary society. Social networking provides online users with a way to share and connect with other online users, and it has permanently changed the lives of individuals, communities, and societies around the world. The world has 1.2 billion social networking users, and online social networking users account for approximately 82% of the Internet users in the world. Facebook is currently the largest social network in the world and the third largest website overall, only after Google and Microsoft sites. In October 2011, the number of Facebook users reached more than half of the global users (about 55%). Facebook is a major social networking site in most countries, but not in every location. Twitter, the first micro-blog social platform, emerged in spring 2009, and the number of its users has grown exponentially. As one of the largest social networks, Twitter now accounts for 1/10 of all Internet users. Social networks have become an indispensable part of present society. The community represents the significant property of real-word social networks as it reflects the relationship between the users. Analyzing network structure and detecting a community of people also play an important part in research on social networks. Detecting a network community structure is of very important theoretical significance and practical value for analyzing network topology structures and network functions and predicting network behavior; it has been widely used in terrorist organizations, organizational structure management, and some other fields. The community reflects the local characteristics of the network of individual behaviors and the relationship between them. It plays a crucial role in the network research for understanding the structure and function of the whole network and can help us analyze and forecast the interaction relationships among elements of the whole network.

The label propagation algorithm (LPA) is a very simple and rapid community detection algorithms [1, 3, 30]. This algorithm is particularly suitable for large social networks with complex communities for various reasons [42]. Although LPA is suitable for a large network, it cannot find overlapping communities, and the division results are highly random. The non-overlapping community means that a node can only divide a community in the community division structure, and the results cannot reflect the true structure of the network. COPRA then extends the LPA and becomes another classical method for detecting overlapping communities [13]. In addition, several other algorithms have been designed to overcome the limitations of the LPA algorithm. For example, SLPA [] and BMLPA [36] alleviate the problem of monster communities by introducing an extra parameter to control the number of labels that a vertex can hold. The core idea of the non-overlapping community detection algorithm based on label propagation is choosing the maximum label value; however, the overlapping community detection has to balance the label strength. If we are too sensitive to label values, it may create monster communities. On the contrary, it will be close to a non-overlapping community.

Facing big data analysis, the parallel processing method arises at a historic moment. Large-scale networks with thousands to millions of nodes are ubiquitous across many different scientific domains. In order to solve the problem of large data and improve the efficiency of the algorithm, parallel computing—especially the Hadoop platform—has attracted more and more scholars' attention.

The MapReduce [17, 22] can be used to achieve distributed clustering and has good scalability and fault tolerance to satisfy the needs of the rapid growth of data. However, in the distributed framework, the clustering algorithm must be operated in a distributed way. Many existing algorithms are not distributed and cannot be easily represented as a single MapReduce process. A parallel clustering algorithm is suggested in [38], which is a parallelized version of DBSCAN [8]. NSLPA [32] used the node similarity parameter *AdjPageSim* to discover community based on label propagation. Konstantin Kuzmin [18] parallelized SLPA, which can significantly speed up the detection of overlapping communities in a large social network.

Motivation Discovering the latent communities is one of the most important research problems in the social network. However, traditional researches on the subject of community detection in this field have some shortcomings: On the one hand, they neglect the meaning of users' influence weight in social network. Most of previous works [3, 4, 9] focus on topological structure or link analysis of the network. In fact, social network contains rich relationship value which can be used to measure the label transferred weight between two nodes. On the other hand, most community detection algorithms [15, 16, 25, 28, 31, 34, 44] are incapable to analyze large-scale dense network.

In this paper, we enhance the LPA by introducing new update and label propagation rules that achieve a higher speed of execution and improve the quality of community detection. Extracting useful knowledge from the modular structure of network data is also a prolific track of research. We focus on the relationship between nodes, whose aim is to identify the label weight value in information spread and detect a community of users usually influenced by the same label.

We summarize the main contribution of this paper as follows: 1) We improve the label propagation algorithm by node confidence with mutual information in a network (i.e., improving the performance of the traditional LPA algorithm by measuring node influence of label updating and changing the label-choosing mechanism when more than one label is contained). 2) We combine the data synchronization and asynchronization to update the label identifier, which can save execution time and avoid label oscillation. 3) We extend the algorithm to fit the parallel processing mechanism for detecting the large-scale data network. 4) The proposed algorithm has the ability to detect overlapping and non-overlapping communities.

The remainder of the paper is organized as follows. In section 2, we review existing approaches to finding communities in network. Section 3 provide some background information on LPA and MapReduce. In section 4, we present the goal of this paper and our proposed method in detail. The experimental evaluation of performance of this algorithm is discussed in section 5. Thus, the paper concludes in section 6.

2 Related works

The problem of finding communities in complex networks is very popular among network scientists, as witnessed by an impressive number of valid works in this field. A huge survey by Fortunato explores all the most popular techniques to find communities in complex networks [9]. Traditionally, a community is defined as a dense subgraph, in which the number of edges among the members of the community is significantly higher than the outgoing edges [4].

Many community detection algorithms have been proposed in the literatures to identity complex community structures in social network. The traditional community refers to a group of nodes in the network with a large similarity, thus forming a close internal connection, while the external sparse population structure. A non-overlapping community means that each node can belong to only one community, and there is no intersection between the communities. In addition to the simple use of similarity to hierarchical cluster, the spectral method was first used to solve the graph segmentation problem. It is widely used in community discovery because of the similarity of the problem [2, 35]. However, the time complexity of spectral method is high, because it involves many matrix eigenvectors calculation, such as similarity matrix. The time complexity of spectral method reaches $O(n^3)$. The concept of random walk has also been successfully introduced into community detection, and many researchers have done a lot of work in this area and got well performance [27, 43]. The basic idea of random walk is that a "random walker" will always walk within a community for a long time due to the high degree of tightness and high connectivity within the community structure. Thus, from one node, the "random walker" will arrive at nodes within the same community in fewer steps, or can define some similarity to further make community discoveries. Another important point is that the random walk approach is easy to extend to community-based discovery of weighted networks, which is one of its advantages. The study of non-overlapping community discovery algorithms is largely due to the pioneering work of Girvan and Newman [10]. The GN community detection algorithm has been cited many times because its research results confirm that this community structure exists in many realistic networks. The Fast Newman algorithm then improved the performance of GN algorithm [24]. Newman and Girvan first introduced the quality function Modularity Q to define a stop criterion to detect community structure. Fast Splitting Algorithm increase the speed of the algorithm by using clustering coefficient instead of edge degree [29]. Some researchers optimize the modularity to find the max modularity of community structure. Mainly include the following categories: Greedy optimization algorithm, Simulated annealing optimization algorithm, Extreme value optimization algorithm, Spectral optimization algorithm and some other optimization algorithms. Although these modularity-based algorithms have been successful in many applications and can find meaningful community structures, the community detection algorithm based on the optimal modularity function cannot get rid of congenital defects due to some properties of the modularity function itself. The process of optimizing modularity may cause the merging small communities without practical significance. Because of the limitation of modularity, more and more scholars in recent years have proposed new methods to solve the problems of community structure detection.

In addition to the methods mentioned above, LPA algorithm is one of the fastest methods for community detection. The LPA used the network structure alone to guide its process and requires neither any parameters nor optimization of the objective function. It starts from a configuration where each node has a distinct label. Each node changes its label to the one carried by the largest number of its neighbors. One drawback of LPA is that it returns different result in different realizations. It caused by the randomness when choosing the multi label case and only consider local minima it reaches. If the node always chooses the label shared by most neighbors at each step, the result could not be always optimal. Gregory [12] applied the similar idea to detection of overlapping

communities. The LPA-S [20] provides the chance to get the global optimal result via selecting some other labels. Barber [1] defines an equivalent objective function based on the number of edges that connect node with labels that penalize the low-quality solution. SLPA [] used speaker-listener model to simulate label propagation with different degree of diversity. WERW-Kpath [23] computed edge centralities in network by random walk to optimize LPA algorithm.

In recent years, parallelized data processing systems are widely used to effectively process large number of data. Following the explosive growth of online social communities, recently the parallel approaches to community detection has been investigated. Reference [17] utilizes the MapReduce framework to develop a scalable parallel non-overlapping community detection method based on infomap. Z Masdarolomoor proposed an agglomerative parallel algorithm using genetic algorithm and local modularity for constructing the communities [22]. Reference [41] proposed a parallel algorithm to find overlapping community structure in directed and weighted complex networks based on the BSP (Bulk Synchronous Parallel) computing framework.Our research not only consider the direct neighbor relationship, but also compute the weight of indirect neighboring users.

3 Preliminaries

Community detection by label propagation belongs to the class of local move heuristics. In previous work, label propagation algorithm is the most common method to detect the community structure and it has approximate linear time complexity []. However, LPA just can find non-overlapping community, so as an extend in COPRA [13], each node updates its labels and the belonging coefficients from the coefficients of all its neighbors in a synchronous manner. SLPA is a general speaker–listener based information propagation process. It spreads label information between nodes according to pairwise interaction rules. In the SLPA, each node has a memory space to store the received information. The probability of observing a label in the memory of a node is perceived as the membership strength [14]. Compared with the existing label propagation methods, our algorithm introduces the concept of confidence to denote the importance of each neighbor in the label updating process.

3.1 Label propagation algorithm

A complex network is modeled by a connected and undirected graph G(V, E), where V is the vertex set and E is the edge set. The label propagation algorithm can be described as below. Each vertex is associated with a label, which is an identifier such as an integer. First, the algorithm initializes the network and each node is given a unique label. Then, every node selects the great number of neighbors' label as itself label. If more than one label is used by the same maximum number of neighbors, one of them is chosen randomly. After several iterations, the same label tends to become associated with all members of a community. Therefore, we can find that LPA has strong random cause it is important for the beginning node. This algorithm can only detect the non-overlapping community structure. Then we will introduce the finding overlapping community with label propagation algorithm.

The algorithm defines a set of community identifiers for each vertex. The COPRA label each vertex x with a set of pairs(c,b), where c is a community identifier and b is a belonging coefficient, indicating the strength of x's membership of community c, such that all belonging coefficient for x sum to 1.

$$b_t(c,x) = \frac{\sum_{y \in N(x)} b_{t-1}(c,y)}{|N(x)|} \tag{1}$$

During each propagation step, the algorithm first constructs the vertex label as above and then deletes the pairs whose belonging coefficient is less than some threshold. They express this threshold as a reciprocal,$1/v$, where v represents the maximum number of communities to which any vertex can belong. If $v < 2$ the algorithm is the same to LPA. In this paper, we proposed an algorithm based on the LPA and the COPRA. However, it is different from the classical algorithm proposed previously.

3.2 Big data processing platform

With the advent of the era of cloud computing, more and more people focus on big data, which brings opportunities for big data processing. Amazon, OpenStack represent virtualization resources which provide users with a large amount of computing and storage resources available anytime and anywhere. The distributed computing and storage environment provides a large scale distributed computing environment such as Hadoop.

Hadoop is a distributed system infrastructure and the distributed storage and distributed computing is the core of distributed system. The most fundamental objective of distributed system design is split the large-scale task into many small tasks [6], and then assign the small tasks to each node with parallel processing, finally generated the results from each processor as the final result. MapReduce is the mainly programming model of implementation the Hadoop architecture. MapReduce [6] as a parallel programming model, is good at dealing with large data and large calculation. The simple MapReduce has three parts: Map function, Reduce function and the main function. If make traditional community detection algorithm parallel with MapReduce programming model and make a good use of cluster computing advantage to handle big users' data, the execution time of the algorithm will be shortening [7].

4 Parallelizing label propagation

The label propagation algorithm has approximated linear time complexity and is very suitable for large network community detecting. As the number of users on social networking has reached the hundreds of millions, using the classical algorithm has caused a high computing complex. If we employ the distributed computing algorithm (i.e., the computing process of the algorithm is distributed) to process the data, the execution time of the community detection algorithm is much more shortened, and the efficiency is also significantly improved. MapReduce, as one of the mainstream parallel computing programming models, is very suitable for processing large-scale data sets. Therefore, it is one of the effective methods for solving the problem of efficiency in the community detecting algorithm.

4.1 Data preprocessing

Paralleling community detection algorithm means distributing data sets into each machine on an average basis. The algorithm is calculated on each machine, the calculation process of each machine is independent, and input data sets are also independent and eventually computed on each machine, thereby generating better results. In the previous discussion, using Hadoop of the MapReduce programming model algorithm's parallelization is a better choice, so here we will utilize Hadoop to parallel the data sets.

In discussing the related work, in the process of synchronous updating, there is a potential oscillation problem that leads hard to the convergence of the algorithm. In an asynchronous update, if a node updates its label value during an iteration, the value must be immediately fed back into all the nodes in the network. This increases the high coupling between the data sets and is contrary to the principles and design of MapReduce's original intention.

As each step of MapReduce involves the process of mapping and reducing, if we design the MapReduce algorithm according to this situation, the mapping process calculates the node update value and the reduction process feeds back the update value to all neighboring nodes. A label value update is required to feed back to all the neighbor nodes that contain this label in the reduction process. If the network contains enormous users and at each node the asynchronous label propagation needs the reduction process, it will require so many reduction processes that it will reduce the efficiency of the algorithm.

In order to solve this problem, we first need to reduce the coupling between the input data sets. In this paper, we will build a structure of input data that contains a node and its neighbor nodes. If we consider the network as the data structure type of node relation, the input set of Mapreduce is the data sets as $<n_i : j, \theta_i(j) + s, \theta_i(s) + \cdots>$. The node j and s are the neighbors of node i. $\theta_i(j)$ means the confidence between node i and node j. We can distribute the input sets averagely to each machine to compute the node confidence and update the label identifier at the same iteration step. As the resulting oscillation and asynchronous updates cost lots of time, we combine a synchronous update with an asynchronous update to evolve the update method. In this paper, we divide the network into n equal subsets. We utilize multiple threads to process a part of the network and feed the results back to the rest of the subsets. This process will be iterated until the whole network is completed. This method solves the oscillation caused by synchrony and the problem of high coupling of the dataset caused by asynchrony. The structure of large social networking is generally sparse [32]; in sparse networks, most of the nodes are not connected to each other, but many nodes can reach another node in a small number of steps. Therefore, the above method combing synchronous and asynchronous updates is feasible.

The mapping step manages a part of the dataset, and the results are fed back to other datasets in the reduction process. Then the output of the Reduce step will be the input for the next Map processing. This processing will repeat until all datasets have been computed, meaning that one iteration execution is completed. The iteration continues until the label value of user nodes reaches the required convergence condition. Figure 1 shows the structure of the processing by MapReduce.

4.2 Label propagation with the confidence between nodes

In the processing of label propagation, the most important part is computing label value. In the classical label propagation algorithm, each node updates its label by replacing it with the label

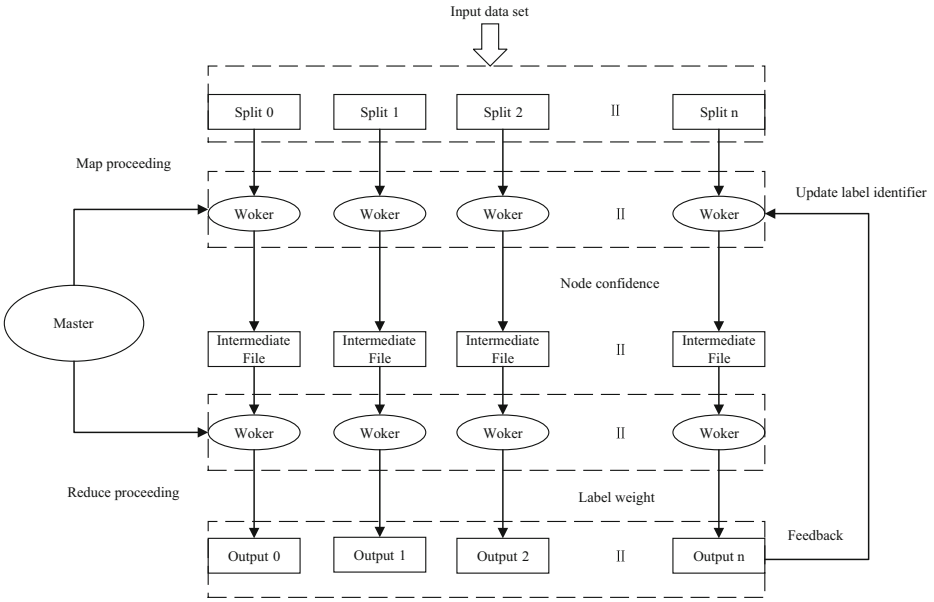

Figure 1 The structure of the processing by MapReduce

used by the greatest number of neighbors. If more than one label is used by the same maximum number of neighbors, one of them is chosen randomly. In this paper, we consider users' relationship as an important impact in the computing of label weight value. There are strong relationships and influence among nodes of the same community. As an analogy to real social phenomena, the users in the same community always share common interests; hence, the users in the same community have more rallying points and influence with each other. However, the influence of different users is different, although they have the same label identification. We calculate the confidence between a pair of nodes as follows.

Definition 1 The confidence of the node v to its neighbor node u. The confidence is defined as

$$\theta_u(v) = \frac{sim_u(v)}{\sum_{i \in N(u)} sim_u(i)} \tag{2}$$

Where $N(u)$ represents the set of neighbors of node u and $sim_u(v)$ represent the similarity between nodes u and v. The Jaccard [43] function is an efficient method for measuring user similarity. The physical meaning of the Jaccard function is very fit for analyzing the similarity of nodes. The intersection of two nodes means the amount of common friends. The union of the neighbors of two nodes means the range in the two users. We apply the Jaccard similarity measure to quantify the similarity between the sets of neighbors for a node. At the same time, we improved the Jaccard function to get more accurate similarity between different nodes.

$$sim_u(v) = \frac{\sum_{x \in N(u) \cap N(v)} I_{ux}}{\sum_{y \in N(u) \cup N(v)} I_{uy}} \tag{3}$$

Where I_{ux} means the mutual information between node u and node x. We use N_i/N to express the probability of the occurrence of vertex i, where N_i is the number of the set containing node

i and its direct neighbors and indirect neighbors in the network. N represents the vertexes amount in the network. P_{ij} means the probability of the occurrence of intersection of node i and node j and themselves. We call n_{ij} the count of the common neighbor vertex i and vertex j and themselves. We define some formulas below.

$$P_i = N_i/N \tag{4}$$

$$P_{ij} = n_{ij}/N \tag{5}$$

$$I_{ij} = P_{ij}\log_2\frac{P_{ij}}{P_i \times P_j} \tag{6}$$

We use the mutual information to show the relationship between neighbors. If the relationship between neighbors is close, the value of mutual information will be large. If the relationship is weak, the value will be close to 0 or even negative. In the simple network model, as in Figure 2, the mutual information value of node 2 and node 5 is negative. Furthermore, node 2 and 5 are obviously not in the same community. Node 2 has the same node confidence value with node 1 and 3 because node 1 and 3 are symmetrical with node 2. The mutual information of the pair of node 2 and 4 is less than I_{12} because node 4 has part of a social relationship with another community. Hence, the idea proposed in this paper to use mutual information to measure the relationship between users is established. We calculate the mutual information value of each direct and indirect neighbor in the simple network model, as shown in Table 1.

In the simple network model, the nodes are divided into two communities intuitively. Nodes 1, 2, 3, and 4 are in the same community while nodes 5, 6, 7, 8, and 9 are in another community. Table 1 indicates that $I_{14} = I_{24} = I_{34} < I_{12}$. This phenomenon is caused by nodes 1, 2, 3, and 4 having a connection with each other, although node 4 connects with another community. Therefore, the influence of node 4 is less than other nodes in the community for node 1. This phenomenon conforms to the community definition: The users have a dense connection with each other and a spare connection with other communities.

For a pair of nodes, we used node confidence to measure the intensity of their connection. For example, as shown in Figure 2, the confidences of node 2 to its neighbors are $\theta_2(1) = 0.387$, $\theta_2(3) = 0.387$,$\theta_2(4) = 0.226$, and $\theta_2(5) = 0$. As shown in Table 1, $I_{25} < 0$ means nodes 2 and 5 belong to two different communities and the label will not transfer between the two nodes. Therefore, we consider the label belonging to

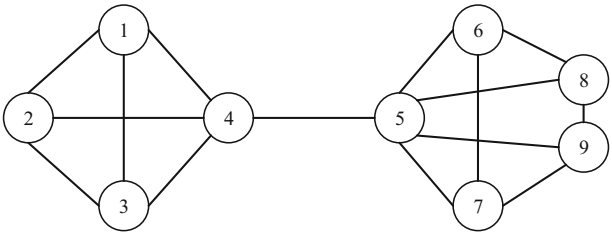

Figure 2 A simple network model

Table 1 Mutual information of every pair of nodes in the simple network

	1	2	3	4	5	6	7	8	9
1	–	0.52	0.52	0.38	−0.16	0	0	0	0
2	0.52	–	0.52	0.38	−0.16	0	0	0	0
3	0.52	0.52	–	0.38	−0.16	0	0	0	0
4	0.38	0.38	0.38	–	−0.16	−0.13	−0.13	−0.13	−0.13
5	−0.16	−0.16	−0.16	−0.16	–	0.26	0.26	0.26	0.26
6	0	0	0	−0.13	0.26	–	0.25	0.25	0.25
7	0	0	0	−0.13	0.26	0.25	–	0.25	0.25
8	0	0	0	−0.13	0.26	0.25	0.25	–	0.25
9	0	0	0	−0.13	0.26	0.25	0.25	0.25	–

node 5 to have no influence on node 2 and define the confidence value as 0 when the mutual information value is 0 or negative between a pair of nodes. If a user is a common node in two communities, the mutual information value will be positive. Therefore, the elimination standard does not mistake the label.

This paper proposes an overlapping community detection algorithm considering the confidence between users called LPAC. As each node has a sequence of its neighbor node ID, we create a sequence corresponding to the confidence between the node and its neighbor node. When the labels propagate, the labels will be sent with node confidence of the two transfer nodes. We sum up the confidence value of neighbor nodes having the same label sent to the target node. $\varepsilon_l(v)$ indicates whether the node v has the label l. If it has this label, the value of $\varepsilon_l(v)$ is 1; otherwise, the value is 0.

$$w_l(i) = \sum_{v \in N(i)} \theta_v(i) * \varepsilon_l(v) \tag{7}$$

We then introduce inflation operation φ_{in} on conference to control the overlapping rate, within which the parameter takes real-number values. After applying φ_{in} to the labels of node i, the belonging coefficient rises to the n^{th} power. The inflation operation φ_{in} is also a normalized method and can be considered the label weight for the node. The inflation operator φ_{in} is defined as:

$$\varphi_{in} w_l(i) = \frac{w_l(i)^{in}}{\sum_{i \in N(i)} w_l(i)^{in}} \tag{8}$$

4.3 Design threshold

In the classical algorithm like LPA and SLPA, the node chooses the label identifier with the maximum value in the label number. Hence, it will find the non-overlapping community with one node holding one label identifier. In some other algorithms, the threshold is set by a constant, which is the better performance on the simulation result.

Our idea was to make the threshold have the actual physical meaning. The retention of labels depends on the degree of node and the current label's amount. A high-degree node may belong to more communities than a low-degree one. If the label weight value is less than $\overline{c(i)}$, the label identifier will be deleted. It means the label has a low impact on the

node, which is less than the average of each label of the node. Hence, the label weight threshold is defined as:

$$\overline{c(i)} = \frac{c(i)}{|N(i)|} \tag{9}$$

where $c(i)$ is the label's amount of node i and $|N(i)|$ is the number of the neighbors (including direct neighbors and indirect neighbors) of node i. If the label weight is bigger than the threshold, the node will keep the label identifier.

Based on the above definitions, the processing of LPAC is as follows:

(1) Initially assign each node in the network a unique label as its own ID, and the confidence value is set to 1.
(2) Calculate the node confidence for every pair of nodes using Eq. (2).
(3) For each node chosen in a random order, update its label from its neighbors (including direct neighbors and indirect neighbors) and calculate the label weight value at the same time by Eq. (8).
(4) Compare the label weight value with threshold as $\overline{c(i)}$. If the label weight value is bigger than threshold, the label identifier will be held. On the contrary, the label identifier will be abandon.
(5) When all labels become stable, stop the algorithm. Else, go back to (3) and repeat the procedure.

The pseudo code describing the LPAC algorithm is reported in Algorithm LPAC:

Input: network graph $G(V, E)$
Output: label identifies of each node $l(i)$

1: **for** i =0 to i <N do
2: $l(i) \leftarrow$ node i _ID
3: $w_l(i) \leftarrow 1$
4: N (i) ← Neighbor (i)
5: **end for**
6: **for** i =0 to i <N do
7: **for** j =0 to j <N do
8: $\theta_i(j) \leftarrow \frac{sim_i(j)}{\sum_{x \in N(i)} sim_i(x)}$
9: **end for**
10: **end for**
11: **while** ($m_t = m_{t-1}$) **do**
12: **for** i =0 to i <N **do**
13: $w_l(i) \leftarrow \sum_{v \in N(i)} \theta_v(i) * \varepsilon_l(v)$
14: **if** $\varphi_{in} w_l(i) > \overline{c(i)}$
15: **return** $l(i)$ for node i
16: **end for**
17: **end while**

The input is the network graph $G(V, E)$, where V is the node set and E is the edge set. The output $l(i)$ is the label identifies of each node i. We defined a set of community identifiers for each node. The terminating condition $m_t = m_{t-1}$ will be introduced as follows.

4.4 Terminating condition

Let the set of community identifiers in the t-th iteration be $i_t = \{c \in V: \exists x \in V(b_t(c,x) > 0)\}$, where V is the set of all nodes in the network. We can know the i_t eventually will reach a minimum value. By the time of $i_t = i_{t-1}$, the iteration should not end unless $|i_t| = 1$, because i_t may decrease again after several propagations. We can get the number of nodes marked by each community identifier $c_t = \left\{(c,i) : c \in V \wedge i = \sum_{x \in V, b_t(c,x)>0} 1\right\}$, but the number often changes in successive iterations, so we cannot expect $c_t = c_{t-1}$. Our approach is to calculate the minimum number of nodes marked by each community identifier beginning with the reduction of the community identifier. The calculating formula is shown in formula (10).

$$m_t = \left\{ \begin{array}{c} (c,x) : \exists p \exists q((c,p) \in c_{t-1} \wedge (c,q) \in c_t \wedge i = min)(p,q), |if\ i_t = i_{t-1} \\ c_t | otherwise \end{array} \right\} \tag{10}$$

where p means the number of nodes labelled with each community identifier at the t-1 th iteration. q means the number of nodes labelled with each community identifier at the t th iteration. When $m_t = m_{t-1}$, we stop the propagation. It is easy to know that it will happen within a finite time, so the algorithm can certainly be terminated. Although we do not prove that we always can get the best results after the t th iterations, the COPRA algorithm termination condition performs well in practice, and the calculation is very simple.

4.5 Parallel community detection with label propagation

The label propagation algorithm is a sequential linear time algorithm for detecting communities. In the parallelized LPAC, we split the network into n partitions of nodes to be processed on p processors. Each processor gets its allocation of nodes that contain a user ID and recreates the network induced by the local node by creating duplicates of nodes. This paper improved the approach to the label update method.

In the Map process, the Map function is used to generate <*key, value*> and update the user ID's new label. It processes a part of the input data, stores label identifiers in temporary files, receives feedback data from the Reduce function, and processes other input data until all input data are processed. Before Map processing, we set the node neighbor and calculate node confidence as input for Map function. In the Map process, the Map function holds the label fed back from the Reduce process.

The Reduce process propagates the label and calculates the label weight using formula (8). By comparing the threshold, the user chooses the label identifiers and feedback to the Map process. If the $\varphi_{in} w_l(i)$ is bigger than the $\overline{c(i)}$, we assign the new label variable *value* to the old label variable *oldlabel* and assign the new label value *weiLabel* to the new label variable *newlabel*. We store the updated node and its new label values in the temporary file.

In the Reduce process, the Reduce function is used to calculate label weight and choose label identifiers. The input of the Reduce function is the output of the Map function. It updates the label value based on the user ID and its new label weight value. The format of output is <key (Node ID), value (new and old label value, neighbor ID, the label value of neighbor)>. If the ratio between the number of the nodes keeps the same new and old labels, the iteration ends; otherwise, the iteration continues.

In the detailed implementation process, we first initialize the user ID and label weight. We then calculate the node confidence and place the user ID with node confidence as the input for the Reduce process. Next, we propagate the label in the Reduce process. Meanwhile, we filter the label identifiers using threshold. We then feed the label with the label weight back to the Map process. If it is temporarily different from the label in the Map process, we update the label value of the neighbor node with the label value corresponding to the user ID in the temporary file. This processing of parallelize label propagation with node confidence called PLPAC as showed in Figure 3.

4.6 Complexity analysis

In this algorithm, initializing every node with unique labels requires $O(n)$ time. The common neighbors' number calculation can be formulated as:

$$\forall n \in V(G), \forall n_i, n_j \in N(n)(i \neq j), Pw(n_i, n_j) \leftarrow Pw(n_i, n_j) + 1 \tag{11}$$

The complexity is $O(n * d^2)$, where d is the average degree of the network. At each node x, we first group the neighbors according to their labels $O(d^2)$. We then calculate the label weight for node x, requiring a worst-case time of $O(d^2)$. This process is repeated at all nodes. In conclusion, the complexity of our LPAC is $O(km^2/n)$ where k is the iteration times. In the spare network, which is often true in the real social network, the complexity can be neat linear $O(n)$.

In the processing of this algorithm, we need store the results of label of each node. The worst-case space of each node is $O(d^2)$. Hence, the space complexity of LPAC is $S(n) = O(n * d^2) = O(m^2/n)$.

5 Performance evaluation

In this section, we first describe the experimental environment and simulation dataset. Then we describe and analysis the experiments that we performed using PLPAC.

5.1 Experiment setup

The language of choice for all implementations is Java according to the JDK 1.6 standard, allowing us to use object-oriented and functional programming concepts while also compiling to native code.

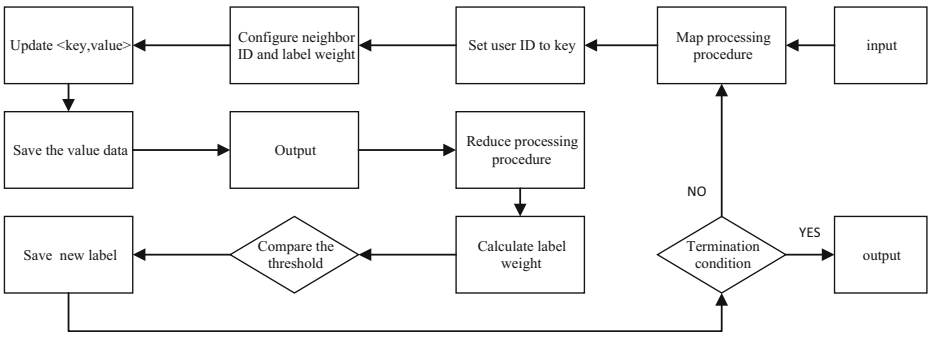

Figure 3 The processing of PLPAC

The Hadoop cluster environment is used in this experiment which consists of 10 machines, a typical master slave mode, (Master-Slaves) structure. The cluster consists of a master node (Master) and four slave nodes (Slave). In the master-slave structure, the main nodes are generally responsible for cluster management, task scheduling and load balancing, and the slave node performs calculation and storage tasks from the main node. For representative experiments, we average quality and speed values over multiple runs in order to compensate for fluctuations. Table 1 provides information on the multicore platform used for all experiments (Table 1).

5.2 Datasets and evaluation metrics

This paper performed experiments on a variety of graphs from different categories of real-world and synthetic data sets. All these networks are treated as unweighted and undirected and are analyzed using classical community detection algorithm. we compare the result of the community detection with some classical algorithms. We can use NMI [5] (Normalized Mutual Information) to measure the performance of parallel LPAC with other algorithms for the known community structure network.

$$NMI(X,Y)=\frac{2I(X,Y)}{H(X)+H(Y)}$$
$$I(X,Y)=\sum_{y\in Y}\sum_{x\in X}p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right)\quad H(X)=-\sum_{i=1}^{n}p(x_i)\log p(x_i) \tag{12}$$

where A and B denote the two partitions of the network. If the found communities are identical to the real communities, then NMI(X, Y) takes its maximum value of 1. If the found communities are totally independent of the real partition, for example when the entire network is classified to be one community, NMI(X, Y) = 0.

For some real networks, there is unknown community structure at present, so this paper will use the EQ function to evaluate the results. The EQ function is an extension of Q function to choose the layer as a community detection result brought out in the tree.

$$EQ=\frac{1}{2m}\sum_i\sum_{v\in Ci,u\in Ci}\frac{1}{O_vO_u}\left(A_{vu}-\frac{k_vk_u}{2m}\right) \tag{13}$$

Where m is the number of the edge, i is the community number, O_v and O_u represent the community number contains node v or u respectively, A is adjacency matrix of the network. In the undirected network, k_v represents the degree of node v. When then community structure is non-overlapping, the value of EQ and Q function is the same.

In the real network part, we use some classical dataset to test and compare the performance of this algorithm with other algorithms. In the artificial network, we can experiment the efficiency of the proposed algorithm. The details of the experimental data sets were showed in Table 3.

Table 2 Operating environment

Environmental category	Describe
Hardware	Intel (R) Xeon () CPU (R), 4G memory
CPU	Intel(R) Xeon(R) E5-2620v3 @ 2.40GHz, 64 threads
Development environment	Eclipse 32, 64bit java version 1.6.0_02

Table 3 datasets

Network	Vertices	Edges	Description
Karate [40]	34	78	Zachary's karate club
Dolphins [11]	62	159	Dolphins social network
Football [21]	115	613	Football American College football
Netsci [26]	1589	2742	Network scientists
Email-Enron [19]	36,692	183,831	Email communication network from Enron
com-Amazon [39]	334,863	925,872	Amazon product co-purchasing network
Artificial network [5]	100 k to 5 M	Mu = 0.1 to 0.8	LFR

5.3 Experimental results

We performed experiments in artificial networks and real networks with several comparison algorithms as follows.

- LPA (Label Propagation Algorithm) [30]: it is a classical fast find community structure algorithm, however, it is only divided non-overlapping communities. We can analysis the overlapping community structure is closer to the reality through the comparison experiments.
- COPRA (Community Overlap PRopagation Algorithm) [13]: it is a classical fast find overlapping community structure algorithm based on LPA. Our research is based on COPRA and further consider the relationship between nodes.
- SLPA (Speakerlistener Label Propagation Algorithm) [37]: it is a Speaker-Listener label propagation algorithm that can handle big scale network data. It considered the nodes as different roles during label propagation to fit for the real information dissemination phenomenon.
- NSLPA (node similarity based label propagation algorithm) []: it is a parallel label propagation algorithm to detect the overlapping community structure. It used PageRank value to help nodes to choose labels during the propagation. We compared with NSLPA to show the effectiveness of node similarity in computing label weight.
- PCOPRA (Parallel Community Overlap PRopagation Algorithm): we parallel the COPRA in our parallel program to show the improvement of the result based on considering the node similarity.

We study the performance of detecting communities of PLPAC. Our experiments are conducted on the generated artificial networks. The resulted NMI values are plotted in Figure 4. We choose SLPA [37], NSLPA [] and parallel COPRA as our comparing algorithms because both are proposed for handling communities based on label propagation. From Figure 4 we can see that PLPAC achieves the highest NMI values among the four clustering algorithms. The results of NSLPA and SLPA are close and are higher than those of PCOPRA. We can find that the performance of SLPA and NSLPA is not identical. There is different label propagation update way between the classical algorithm (like SLPA) and the parallelized algorithms (like NSLPA). The synchronization mechanism is necessary for designing the parallel steps of the algorithms. On the contrary, COPRA and SLPA update the labels asynchronously. Our proposed algorithm combined synchronization and asynchronization has better performance in varying node number networks. It means the improvement of update mechanism is successful.

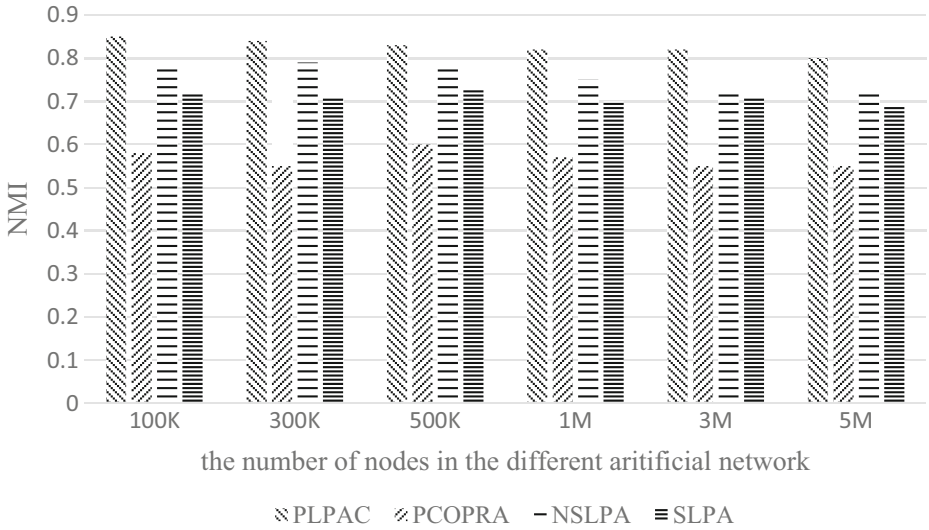

Figure 4 The NMI value in artificial network

In the Figure 5 illustrates how running time varies with the increasing of network scale. Clearly, the total running time includes the time spent on communication between processors and time spent on execution of the algorithm itself. It is obvious that the time cost of all the algorithms increases nearly linearly with network size. When the number of network nodes in thousands of counting, the speedup of the other algorithms caused by parallel computation is not evident. It possible caused by the time spent on data processing is comparable with the time spent on cluster administration and communication. However, when the number of network nodes in millions of counting, the speedup of the parallelized algorithms shows the advantage than classical algorithms. Parallel computation becomes remarkable when network scale increases beyond the capability of

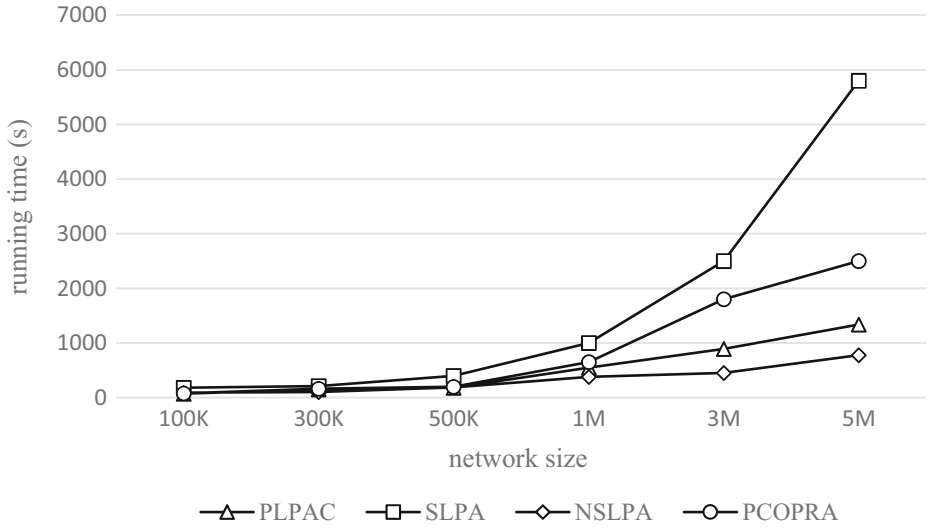

Figure 5 The run time in artificial network

a single-machine algorithm. SLPA and NSLPA run faster than PCOPRA because the speak and listen strategy is simpler than the label updating strategies used in PLPAC. NSLPA exhibits better scalability than SLPA, which is largely due to the parallel speak and listen scheme. Although the run time of PLPAC is higher than NSLPA, the high NMI value shows the algorithm this paper proposed can detect the better communities.

The experimental results for the real networks are presented in Figure 6. The Figure 6 shows the performances of the algorithms on the real networks are considerably different from those on the artificial networks. The networks known the structure can utilize NMI to show the performance of community detection. The NMI value is higher, the structure divided is closer to the real structure. We can find that the algorithm proposed by this paper has the best performance. It means the community divided is closer to real community structure than other results. In Figure 6, with the node number of network raise, the result divided by LPA is poor. It because the traditional label propagation algorithm cannot handle big data network. The results of COPRA are better than LPA because it allows node belongs to more than one communities. Thus, the overlapping community structure is fit for the real-world situation. The different results between NSLPA and SLPA are caused by the similarity proposed by NSLPA. Hence, we parallelize the label propagation algorithm and consider nodes confidence to fit the modern large-scale networks.

In order to detect the performance of the proposed algorithms in large data networks, we used Email-Enron and com-Amazon datasets with unknown community structure to detect the performance. Hence, we utilize EQ function to measure the accuracy of the divided result with big data network.

We show the performance with different algorithms in Figure 7. In the Karate network, the small size of the network and its limited structure cause a low value of modularity value. However, the 0.416 is the highest among the results detected by many algorithms. In Dolphins and Football networks, the performance of each algorithm performs very well caused by the clear community structure in the networks.

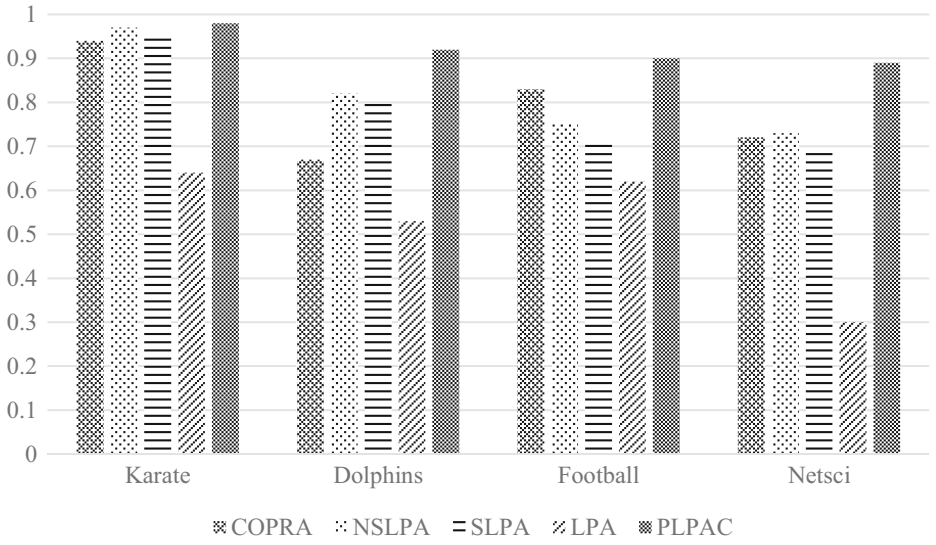

Figure 6 The NMI value in real network

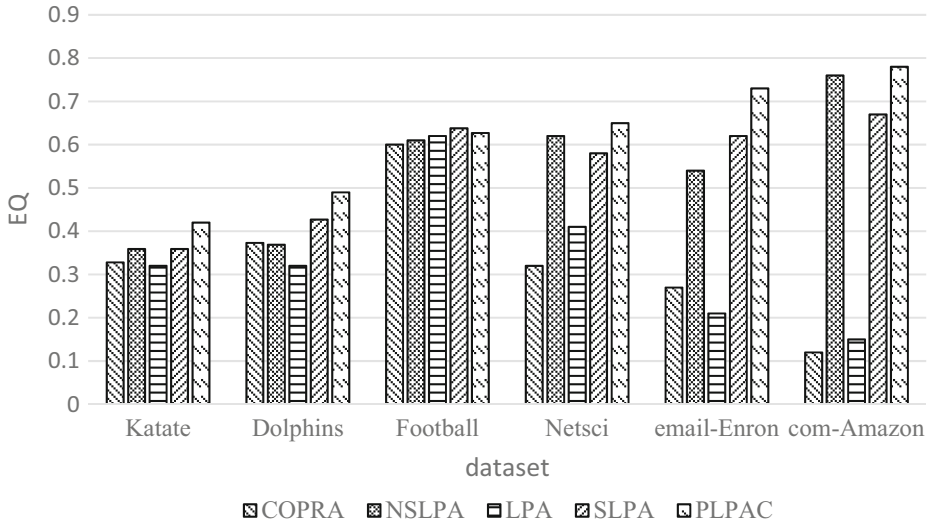

Figure 7 The EQ value in real network

With the increasing complexity of network relations, the advantages of parallel algorithms become significant. We can see that in the com-Amazon with 334,863 nodes as a big data network the traditional algorithm LPA [30] and COPRA [13] have the poor community division. The NSLPA can divided community better than SLPA, which is parallel the label propagation proceeding to fit for large scale network. The PLPAC gets high score in EQ function comparing with other algorithms and performs well in every size network.

We showed the running time in Figure 8 including each step and total running time. We do not show the running time of dividing by Karate, Dolphin and Football dataset,

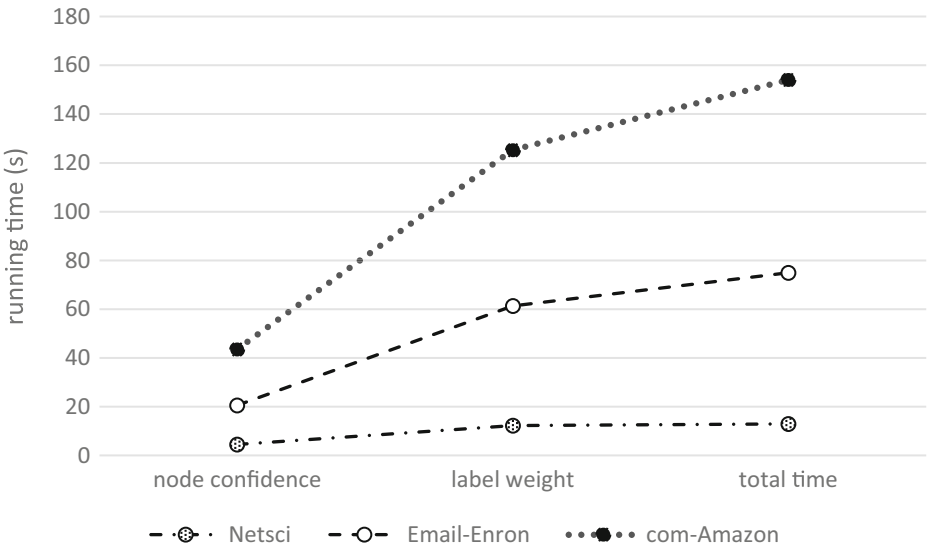

Figure 8 The running time of each step

because the number of nodes is too small for the parallel system. We can find the calculation of node confidence cost more time with the number of network raise. Meanwhile, we find the execution time of node confidence is only a small proportion of the total running time. The main running time are cost by label propagation with iteration repeatedly. The total time not only contains the node confidence and label weight time. It also contains the time cost by communication and management among the machines.

In Figure 9, we can find the communities number is reduced with the iteration of label propagation. In order to intuitive analysis the convergence of the proposed algorithm, we used Zachary karate club network to show the community detection performance with the iteration by LPAC. We find the NMI value becomes larger with the convergence of the number of communities. In our algorithm, we divided this network need 4 iterations, and the 5th iteration means the communities numbers reaches a steady state.

At the end of each run, we calculated the total execution time and speedup using formula shown in (14), efficiency according to (15).

$$Speedup = \frac{T_1}{T_n} \tag{14}$$

Where, T_1 means the running time on the single machine, the T_n means the running time on the cluster.

$$Efficiency = \frac{Speedup}{p} \tag{15}$$

Experiments are conducted on the 1 M network with a varying number of machines to evaluate the effect of cluster scale on the performance. Figure 10 shows that a boost on running speed caused by adding machines to the cluster is evident. As the number of processors increase, the growth rate of speedup is decay. Because the time cost by communication and management among the machines will rise with more machines. Therefore, we should balance the number of the processors to detect the accurate community structure in the shortest time.

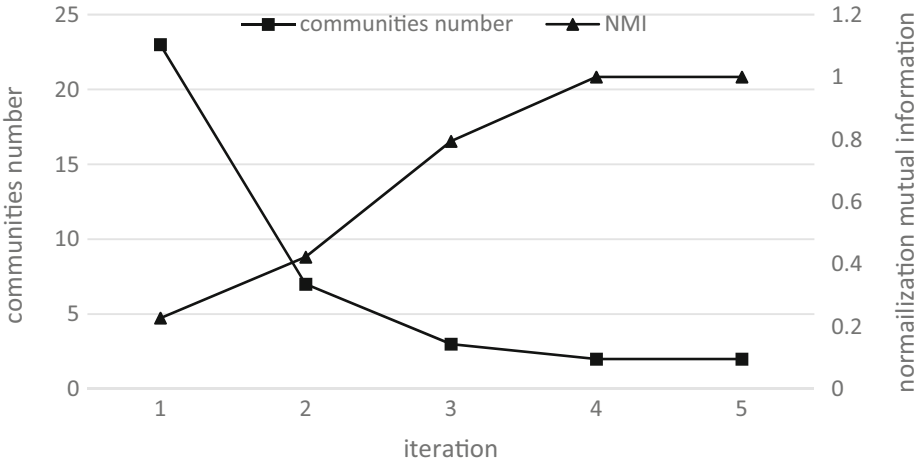

Figure 9 The communities number and NMI on Zachary's karate club network with different iterations

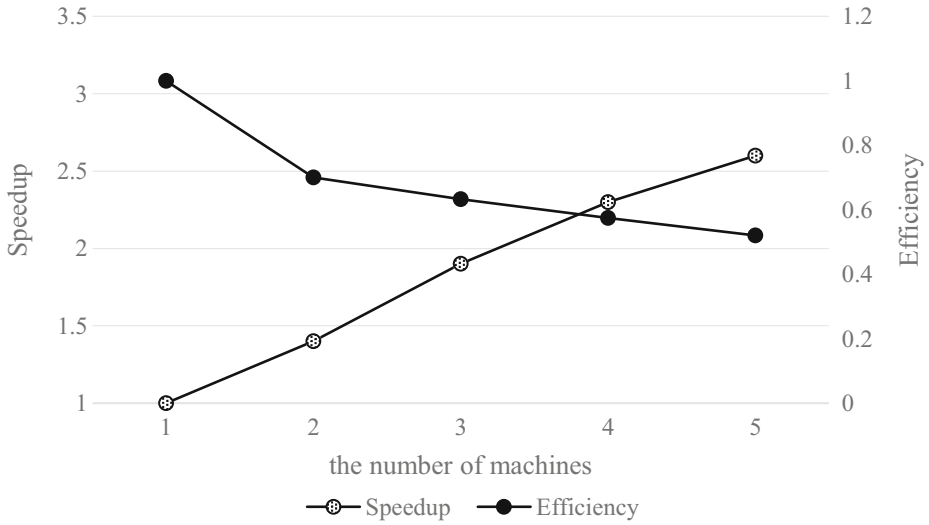

Figure 10 Speedup and efficiency for a network

6 Conclusion and future work

In this paper, we proposed a parallel label propagation algorithm with node confidence to detect communities. In addition, we evaluated the performance of a multi-threaded parallel implementation of the label propagation algorithm and demonstrated that using a modern multiprocessor can significantly reduce the time required to analyze the structure of different networks and output communities. We found that, with the increasing number of processors, the rate of speedup reduces slowly. This can be explained by the fact that more and more communication time is spent on the processors, which should be considered. The implementation of PLPAC proves that it can detect the communities in large networks with high accuracy. Compared with other algorithms, the simulation result shows that our algorithm can correctly identify overlapping community structures from real data, and the improved label propagation with node confidence is very effective. Finally, the speedups on various datasets and different numbers of machines are satisfactory.

In our future work, we plan to raise a greater number of the processors and evaluate the experimental performance. In addition, we will explore other parallel programming paradigms and compare their performance with our parallel approach.

Acknowledgments This work has been supported by the Fundamental Research Funds for the Central Universities 2016YJS029 and the National Natural Science Foundation of China under Grant 61401015. Academic Discipline and Postgraduate Education Project of Beijing Municipal Commission of Education.

References

1. Barber, M.J., Clark, J.W.: Detecting network communities by propagating labels under constraints. Phys. Rev. E Stat. Nonlinear Soft Matter Phys. **80**, 026129 (2009)
2. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. Adv. Neural Inf. Proces. Syst. MIT Press. **14**, 585–591 (2001)
3. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E. **70**, 066111 (2004)

4. Coscia, M., Rossetti, G., Giannotti, F., Pedreschi, D.: Demon: a local-first discovery method for overlapping communities. 615–623 (2012)
5. Danon, L., Dazguilera, A., Duch, J., Arenas, A.: Comparing community structure identification (2005)
6. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM. **51**, 107–114 (2008)
7. Dean, J., Ghemawat, S.: MapReduce: a flexible data processing tool. Commun. ACM. **54**, 72–77 (2010)
8. Ester, M., Kriegel, H. P., Xu, X.: A densitybased algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In International Conference on Knowledge Discovery and Data Mining 226–231, (1996)
9. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**, 75–174 (2009)
10. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. **99**, 7821–7826 (2002)
11. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. U. S. A. **99**, 7821–7826 (2002)
12. Gregory, S.: Finding overlapping communities in networks by label propagation. New J. Phys. **12**, 2011–2024 (2001)
13. Gregory, S.: Finding overlapping communities in networks by label propagation. New J. Phys. **12**, 2011–2024 (2010)
14. He-Li, S., Jian-Bin, H., Yong-Qiang, T., et al.: Detecting overlapping communities in networks via dominant label propagation. Chin. Phys. B. **24**, 551–559 (2015)
15. Hu, Y., Li, M., Zhang, P., Fan, Y., Di, Z.: Community detection by signaling on complex networks. Phys. Rev. E. **78**, 016115 (2008)
16. Jaccard, P.: The distribution of the Flora in the alpine zone. New Phytol. **11**, 37–50 (1912)
17. Jin, S., Yu, P.S., Li, S., Yang, S.: A parallel community structure mining method in big social networks. Math. Probl. Eng. **2015**(10), 1–13 (2015)
18. Konstantin Kuzmin, S., Shah, Y., Szymanski, B. K.: Parallel overlapping community detection with slpa. In International Conference on Social Computing 204–212 (2013)
19. Leskovec, J., Lang, K.J., Dasgupta, A., et al.: Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. Internet Math. **6**, 29–123 (2009)
20. Li, S., Lou, H., Jiang, W., et al.: Detecting community structure via synchronous label propagation. Neurocomputing. **151**, 1063–1075 (2015)
21. Lusseau, D., Boisseau, S.K., et al.: The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. Behav. Ecol. Sociobiol. **54**, 496–405 (2004)
22. Masdarolomoor, Z., Azmi, R., Aliakbary, S., Riahi, N.: Finding community structure in complex networks using parallel approach. In Ifip International Conference on Embedded and Ubiquitous Computing 474–479 (2011)
23. Meo, D., Pasquale, F., et al.: Enhancing community detection using a network weighting strategy. Information Sciences an International Journal. **222**, 648–668 (2013)
24. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Phys. Rev. E. **69**, 066–133 (2004)
25. Newman, M.E.J.: Modularity and community structure in networks. Proc. Natl. Acad. Sci. **103**, 8577–8582 (2006)
26. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E. **74**, 036104 (2006)
27. Pons, P., Latapy, M.: Computing communities in large networks using random walks (long version). J. Graph Alg. App. 10(2):284–293 (2005)
28. Pons, P., Latapy, M.: Computing communities in large networks using random walks. J. Graph Algorithms Appl. **10**, 191–218 (2006)
29. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proc. Natl. Acad. Sci. U. S. A. **101**, 2658–2663 (2004)
30. Raghavan, U., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E. **76**, 036106 (2007)
31. Schuetz, P., Schuetz, P., Caflisch, A.: Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. Phys. Rev. E Stat. Nonlinear Soft Matter Phys. **77**, 046–112 (2008)
32. Song, Q., Li, B., Yu, W., Li, J., Shi, B.: Nslpa: A node similarity based label propagation algorithm for real-time community detection. In Ieee/acm International Conference on Utility and Cloud Computing 896–901 (2014)
33. Ugander, J., Backstrom, L.: Balanced label propagation for partitioning massive graphs. In ACM International Conference on Web Search and Data Mining 507–516 (2013)
34. Wakita, K., Tsurumi, T.: Finding community structure in mega-scale social networks. In: International Conference on World Wide Web. ACM, 1275–1276 (2007)

35. White, S., Smyth, P.: A spectral clustering approach to finding communities in graph. In Siam International Conference on Data Mining (2005)
36. Wu, Z.H., Lin, Y.F., Gregory, S., Wan, H.Y., Tian, S.F.: Balanced multi-label propagation for overlapping community detection in social networks. J. Comput. Sci.Technol. **27**, 468–479 (2012)
37. Xie, J., Szymanski, B. K., Liu, X.: Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In IEEE International Conference on Data Mining Workshops 344–349 (2012)
38. Xu, X., Jäger, J., Kriegel, H.P.: A fast parallel clustering algorithm for large spatial databases. Data Min. Knowl. Disc. **3**, 263–290 (1999)
39. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. Knowl. Inf. Syst. **42**, 181–213 (2015)
40. Zachary, W.W.: An information flow model for conflict and fission in small groups. J. Anthropol. Res. **44**, 452–474 (1997)
41. Zhang, J., Ge, S.: A parallel algorithm to find overlapping community structure in directed and weighted complex networks. In Second International Conference on Instrumentation, Measurement, Computer, Communication and Control 1561–1564 (2013)
42. Zhang, Q., Qiu, Q., Guo, W., et al.: A social community detection algorithm based on parallel grey label propagation. Comput. Netw. **107**, 13–143 (2016)
43. Zhou, H.: Distance, dissimilarity index, and network community structure. Phys. Rev. E. **67**, 061–901 (2003)
44. Zhou, H., Lipowsky, R.: Network brownian motion: a new method to measure vertex-vertex proximity and to identify communities and subcommunities. Lect. Notes Comput. Sci. **3038**, 1062–1069 (2004)