Computer Networks 000 (2016) 1–11

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

A social community detection algorithm based on parallel grey label propagation

Qishan Zhang[a], Qirong Qiu[a], Wenzhong Guo[b,c,d], Kun Guo[b,c,d,*], Naixue Xiong[e]

[a] School of Economics and Management, Fuzhou University, Fuzhou, 350116, China
[b] College of Mathematics and Computer Science, Fuzhou University, Fuzhou, 350116, China
[c] Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou, 350116, China
[d] Key Laboratory of Spatial Data Mining & Information Sharing, Ministry of Education, Fuzhou 350002, China
[e] Department of Business and Computer Science, Southwestern Oklahoma State University, OK 74074, USA

ARTICLE INFO

Article history:
Received 30 October 2015
Revised 28 May 2016
Accepted 2 June 2016
Available online xxx

Keywords:
Community detection
Parallel computation
Label propagation

ABSTRACT

Community detection is one of the important methods for understanding the mechanism behind the function of social networks. The recently developed label propagation algorithm (LPA) has been gaining increasing attention because of its excellent characteristics, such as a succinct framework, linear time and space complexity, easy parallelization, etc. However, several limitations of the LPA algorithm, including random label initialization and greedy label updating, hinder its application to complex networks. A new parallel LPA is proposed in this study. First, grey relational analysis is integrated into the label updating process, which is based on vertex similarity. Second, parallel propagation steps are comprehensively studied to utilize parallel computation power efficiently. Third, randomness in label updating is significantly reduced via automatic label selection and label weight thresholding. Experiments conducted on artificial and real social networks demonstrate that the proposed algorithm is scalable and exhibits high clustering accuracy.

1. Introduction

As Web 2.0 applications rapidly develop, social networking has become an important means for people to share information and communicate with each other worldwide. Aside from analyzing network structures, detecting groups of people with common interests or behavior patterns also plays an important role in research on social networks. Unlike traditional networks, social networks are complex and have their own special characteristics. First, social networks are usually extremely large in scale. For example, active Facebook users have already reached one billion in 2014 [1]. The high number of vertices and edges of networks significantly affect processing capability. Second, social networks are typical heterogeneous networks that consist of different types of data from various sources. For example, people can share their favorite writings, pictures, songs, or movies on Facebook or Twitter. Therefore, handling heterogeneous data is a necessary function of modern tools for social network analysis. Finally, community structures hidden in social networks may be highly complex. Communities may overlap or may be hierarchical. For example, the circle of friends of a person is usually combined with those of his/her colleagues, or teachers working in a university can be grouped into different levels, such as colleges and departments.

Many community detection algorithms have been proposed to identify complex community structures in social networks [2]. Community detection appears to be similar to traditional clustering or graph partitioning. Thus, several effective clustering or graph partitioning algorithms have been applied in community detection. The Kernighan–Lin algorithm aims to minimize the difference between intra-edges and inter-edges to detect communities [3]. Spectral methods determine the minimum cut to separate a graph and recursively divide a network into sections where communities emerge naturally [4]. The critical modularity character of networks was discovered when hierarchical clustering was first applied by Grivan and Newman to detect communities [5]. Since its discovery, modularity has been used frequently to design community detection algorithms. For example, it can be utilized as an evaluation measure, an optimization objective, or a stopping criterion [2].

Early community detection algorithms cannot handle large social networks because these algorithms exhibit space complexity and require substantial time [1]. A promising algorithm, called the label propagation algorithm (LPA), was proposed recently [6]. This algorithm is particularly suitable for large social networks with complex and overlapping communities because of various

* Corresponding author.
E-mail addresses: gukn123@163.com, gukn@fzu.edu.cn (K. Guo), xiongnaixue@gmail.com (N. Xiong).

http://dx.doi.org/10.1016/j.comnet.2016.06.002
1389-1286/

reasons. First, the running time of LPA is linear to network size. Thus, it can process extremely large networks. Second, LPA does not require predefining the number of communities. Communities emerge gradually during the propagation of labels. Finally, the algorithm allows a vertex to carry multiple labels [7]. It finds overlapping communities naturally. However, LPA also has limitations. First, vertex labels are initialized randomly, which results in the unstable performance of LPA. Second, the greedy label updating strategy tends to produce extremely large communities (i.e., monster communities) that swallow up most small communities near them. Recently, several algorithms have been proposed to address the limitations of the LPA algorithm The COPRA) [7], SLPA [8], and BMLPA [9] algorithms alleviate the problem of monster communities by introducing an extra parameter to control the number of labels that a vertex can hold. To deal with extremely large networks, SLPA has been further extended to the MPI model [10] to run parallel on a multi-core computer [11]. The PSCAN algorithm [12], which was designed on the Hadoop framework, provides another parallel scheme for the MapReduce model [13]. The LPA–CNP algorithm [14] presents a new label updating strategy by considering the effect of neighbors that are more than one hop away. The CK–LPA algorithm [15] provides another label updating strategy that weighs the label of a vertex according to whether the vertex is in the community kernel. The LPA–S algorithm [16] suggests that a synchronous label propagation strategy is more effective than the default asynchronous strategy. In our previous works [17, 18], we proposed a parallel affinity propagation technique that was efficient in discovering communities in large social networks. We also extended LPA under the MapReduce framework to process large networks by considering the merits of this algorithm [19]. In the current study, we design a new LPA parallelization scheme from a different perspective, which addresses the limitations of our previous work [19]. The major contributions of this study are summarized as follows.

(1) The neighbors of a vertex and the similarity distribution of the neighbors are considered in measuring similarity among vertices by including grey relational analysis in calculating similarity.
(2) Jaccard similarity is integrated into grey similarity. Thus, the neighbors of a vertex and the neighbors of its neighbors are considered in calculating similarity.
(3) A new parallel similarity calculation scheme based on the MapReduce model is developed to improve the calculation of similarity among vertices.
(4) Vertex labels are updated according to both weight thresholding and automatic selection strategies.
(5) New parallel operators of the Spark framework are adopted in the proposed algorithm to improve its scalability.

Comprehensive experiments conducted on both artificial and real social networks verify the effectiveness and efficiency of the new algorithm.

The remainder of the paper is organized as follows. In Section 2, the theoretical basis of the grey relational analysis is introduced. Section 3 provides a comprehensive discussion of the concept behind the proposed algorithm, its detailed implementation, and its complexity analysis. The results of the experiments conducted on artificial and real social networks are presented in Section 4. Section 5 concludes the study and presents an outline for future research.

2. Grey relational analysis

Grey relational analysis is one of the important components of grey methods [20]. Unlike statistical and fuzzy methods, grey methods do not require large samples or a distribution hypothesis. Instead, analysis is performed by regarding problem objects as a whole, and thus, this method is suitable for data with complex intrinsic structures. Recently, Zhang introduced entropy theory of information sequence to improve grey relational analysis [21]. The key concepts of grey relational analysis are defined as follows.

Definition 1. Let $X = \{\vec{x}_i | \vec{x}_i = (x_i(1), x_i(2), \ldots, x_i(n)), i = 0, 1, \ldots, m\}$ be a set of sequences whose lengths are all equal to n. $\vec{x}_0$ is called the reference sequence, and the remaining $\vec{x}_i$s are called the comparison sequences. Given any real number $\xi \in [0, 1]$, let

$$\gamma(x_0(j), x_i(j)) = \frac{\min_i \min_j |x_0(j) - x_i(j)| + \xi \max_i \max_j |x_0(j) - x_i(j)|}{|x_0(j) - x_i(j)| + \xi \max_i \max_j |x_0(j) - x_i(j)|}, \quad (1)$$

then $\gamma(x_0(j), x_i(j))$ satisfies four grey relational axioms, namely normalization, even symmetry, integrity and closeness. $\gamma(x_0(j), x_i(j))$is called the grey relational coefficient of $\vec{x}_i$ to $\vec{x}_0$ at the jth point. ξ is called the differentiation coefficient. Let

$$\gamma(\vec{x}_0, \vec{x}_i) = \frac{1}{n} \sum_{j=1}^{n} \gamma(x_0(j), x_i(j)), \quad (2)$$

then $\gamma(\vec{x}_0, \vec{x}_i)$ is referred to as the grey relational degree of $\vec{x}_i$ to $\vec{x}_0$.

Definition 2. Let $\vec{r}_i = (\gamma(x_0(j), x_i(j)))$ be the grey relational coefficient sequence of the ith comparison sequence. The mapping

$$\begin{gathered} \phi : \vec{r}_i \to \vec{v}_i \\ \gamma(x_0(j), x_i(j)) \to v(x_0(j), x_i(j)) \\ v(x_0(j), x_i(j)) = \frac{\gamma(x_0(j), x_i(j))}{\sum_{j=1}^{n} \gamma(x_0(j), x_i(j))} \end{gathered} \quad (3)$$

is called the distribution mapping of the grey relational coefficient. $v(x_0(j), x_i(j))$ is the grey relational density of the jth point in the ith comparison sequence. All the grey relational densities of this comparison sequence form a grey relational density sequence, which is denoted as $\vec{v}_i$.

Definition 3. Let $V = \{\vec{v}_i | i \in N\}$ be the grey relational density sequence set. Thus, function

$$I(\vec{v}_i) = -\sum_{j=1}^{n} v(x_0(j), x_i(j)) \ln v(x_0(j), x_i(j)) \quad (4)$$

is called the grey relational entropy of the ith comparison sequence.

Definition 4. Let $I(\vec{v}_i)$ be the grey relational entropy of the ith comparison sequence and I_m be the largest grey relational entropy of the grey relational coefficient sequence. Then,

$$E(\vec{x}_0, \vec{x}_i) = I(\vec{v}_i)/I_m \quad (5)$$

is called the entropy relational degree of the ith comparison sequence.

Definition 5. Let $\gamma(\vec{x}_0, \vec{x}_i)$ and $E(\vec{x}_0, \vec{x}_i)$ be the grey relational degree and the entropy relational degree of the ith comparison sequence, respectively. Then,

$$B(\vec{x}_0, \vec{x}_i) = E(\vec{x}_0, \vec{x}_i) \times \gamma(\vec{x}_0, \vec{x}_i) \quad (6)$$

is called the balanced closeness degree of the ith comparison sequence.

Please cite this article as: Q. Zhang et al., A social community detection algorithm based on parallel grey label propagation, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.06.002

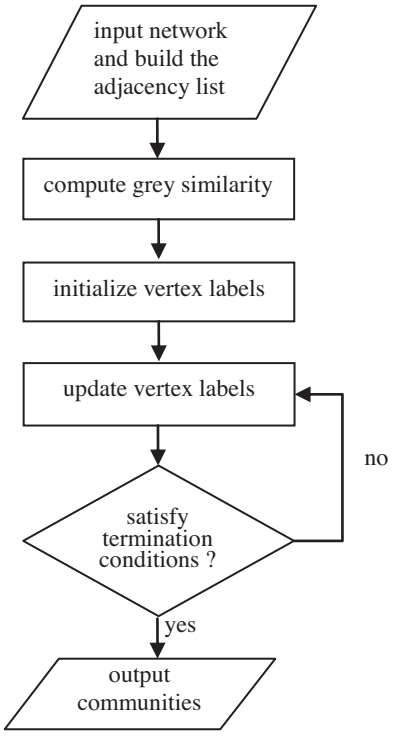


Fig. 1. Procedures of grey label propagation.

The balanced closeness degree considers both the closeness of the points from grey relational factor sequences and global indiscriminate closeness. Thus, the strong associative inclination of the local data points can be addressed.

3. Parallel grey label propagation

3.1. Grey label propagation

Each vertex in LPA updates its label according to the labels of its neighbors. The labels chosen by more neighbors are likely to be selected as the new labels of a vertex [6–9]. This situation indicates that the labels of each neighbor are equally weighted from the perspective of label weighting. However, different neighbors may have varying effects on a vertex [22]. Hence, weighting different labels carried by various neighbors is preferable. Accordingly, we select the new labels of a vertex in three steps. First, the influence of each neighbor is measured through its similarity with the vertex. Subsequently, the labels of a neighbor are weighted through its influence. Finally, labels with high weights are prioritized as the new labels of a vertex. The grey relational degree can be viewed as a composite metric that measures the similarity of two sequences not only in a pointwise way like the Euclidean distance but also in a probability distribution way like the Kullback–Leibler divergence. Moreover, by combining it with the Jaccard index, the grey relational degree can measure the similarity according to set coverage as well. Therefore, we adopt this metric to calculate the similarity of the vertices and call the new label propagation process grey label propagation.

The general procedures of grey label propagation are illustrated in Fig. 1.

3.2. Parallelization of grey label propagation

The parallelization of grey label propagation includes the parallelization of four critical steps: adjacency list construction, grey similarity calculation, vertex label initialization, and vertex label updating.

3.2.1. Parallel adjacency list construction

When a network is too large to be stored on the memory of a single machine, it is typically saved in the distributed memory, such as the HDFS (Hadoop Distributed File System) or HBase. Thus, a parallel method can be developed to read the network from the distributed memory and build an adjacency list. A simple and effective scheme for constructing a parallel adjacency list is depicted in Fig. 2. First, each edge e_{ij} is read from the distributed memory and mapped onto a tuple with i as the key and j as the value. Then, the tuples are grouped by the keys. The values with the same key i are reduced into a set $NB(i)$, including all the neighbors of i, namely, $NB(i) = \{j | e_{ij} \text{isanedge}\}$. After all the $NB(i)$s are built, the adjacency list of the network is constructed.

3.2.2. Parallel grey similarity calculation

Vertices generally do not have the same number of neighbors. However, the balanced closeness degree requires all the sequences to have equal lengths. Thus, developing a method to transform the neighbors of two vertices into two lists with equal lengths for grey similarity calculation is necessary. A feasible scheme is presented in Fig. 3. First, the neighbor lists of a vertex i and its neighbor j, i.e., $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_m)$, are used to compute Jaccard similarity. Second, the lists of vertices i and j of Jaccard similarity, i.e., $(s_{x1}, s_{x2}, \ldots, s_{xn})$ and $(s_{y1}, s_{y2}, \ldots, s_{yn})$, are aligned and padded with zeroes to build new similarity lists with equal lengths. In particular, the new similarity list of vertex i will be padded with zero if a vertex k is its neighbor and vice versa if it is not a neighbor of vertex j. Hence, two transformed lists with equal lengths are ready for calculating balanced closeness degree according to Eq. (6) . Furthermore, the distribution of neighbors and their neighbors is regarded as a whole, which is deemed beneficial for label selection [14].

A naive scheme for calculating parallel grey similarity is depicted in Fig. 4 [12]. The main idea is to collect the neighbor lists of two vertices to calculate their Jaccard similarity. A tuple with key (i, j) and value $NB(i)$ is sent for each neighbor j of vertex i in $NB(i)$ using the *map* operator. The first *reduce* operator will receive both $NB(i)$ and $NB(j)$ for the same key (i, j) to support the calculation of Jaccard similarity $s(i, j)$. The second *reduce* operator is used to collect the Jaccard similarity $s(i, j)$ of all the neighbors of vertex i to build similarity vector $\mathbf{s}_i$.

However, the naive scheme has a serious defect. Fig. 4 shows that the *map* operator sends $NB(i)$ for each neighbor of vertex i. Hence, this scheme requires $O(k^2)$ space for each vertex and $O(nk^2)$ for the entire network, where k is the average number of neighbors of a vertex. For networks with a smooth degree distribution, the space required can be reduced to $O(n)$ because k is typically significantly smaller than n. By contrast, for networks with a skewed degree distribution, space complexity may reach $O(n^3)$ because of the extremely large k for a few vertices. Therefore, the scheme may be unable to handle a moderate network if its degree distribution is highly skewed. The defect is caused by the excessive propagation of the same $NB(i)$, i.e., broadcasting the same neighbor list of a vertex too many times. To address this problem, we design a scheme that broadcasts $NB(i)$ to each node of a computation cluster only once. A traditional parallel computation framework, such as Hadoop, does not support the direct transfer of information to a certain node. However, the parallel computation framework Spark can effectively support orientated data transfer with the aid of the new *partitionBy* and *mapPartition* operators. Thus, a new efficient scheme for parallel similarity calculation is developed (Fig. 5).

Please cite this article as: Q. Zhang et al., A social community detection algorithm based on parallel grey label propagation, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.06.002

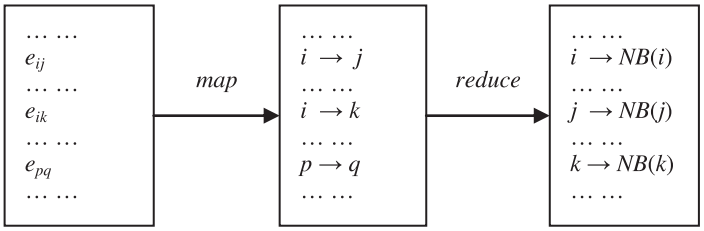


Fig. 2. Scheme for constructing a parallel adjacency list.

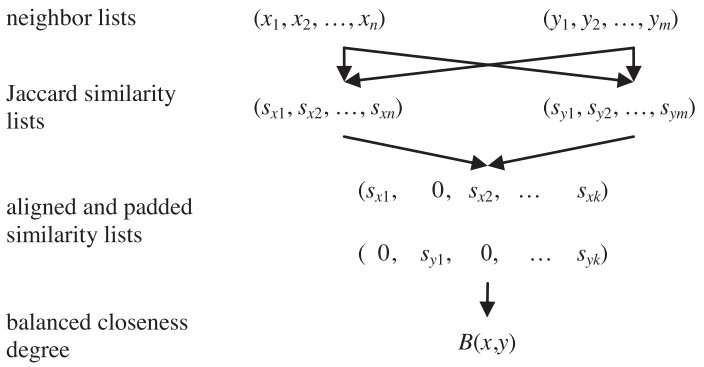


Fig. 3. Scheme of grey similarity calculation.

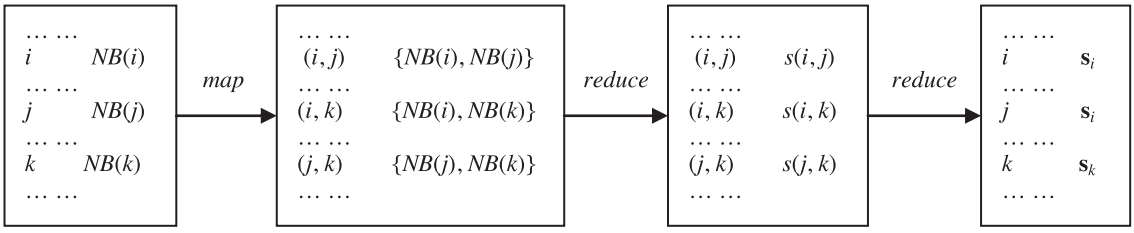


Fig. 4. Naive scheme for parallel similarity calculation.

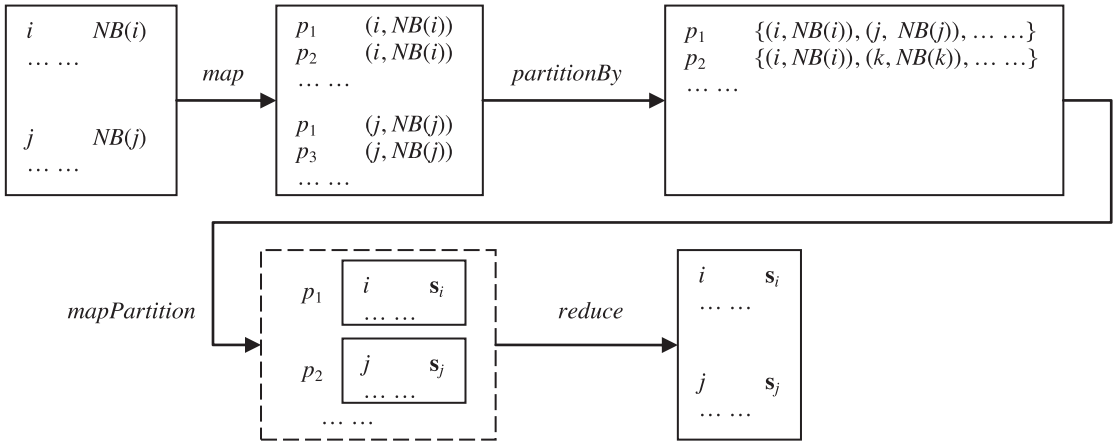


Fig. 5. New scheme for parallel similarity calculation.

First, the partition id p_k for each of its neighbor is calculated for any vertex i. Second, the tuple $(i, NB(i))$ is sent to each partition only once. Third, by using the *partitionBy* operator, each partition will have sufficient vertices, along with their neighbors, for similarity calculation, i.e., for each k in $NB(i)$, $NB(k)$ is also in the same partition. Then, the similarity scheme depicted in Fig. 3 is adopted to calculate similarity vector $\mathbf{s}_i$ for each vertex i in the partition by using the *mapPartition* operator. Finally, all the similarity vectors are collected using the *reduce* operator. Determining that N_p copies of the same $(i, NB(i))$ will exist at most is easy. Thus, space occupation can be significantly reduced.

3.2.3. Parallel vertex label initialization

The parallelization of vertex label initialization is a straightforward process, as shown in Fig. 6. Vertex i is initialized with a label set $L(i)$, which contains only itself as its label. The weight of each label is initialized to 1.0. Our algorithm can identify the overlapping communities by allowing each vertex to carry a label set.

Please cite this article as: Q. Zhang et al., A social community detection algorithm based on parallel grey label propagation, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.06.002

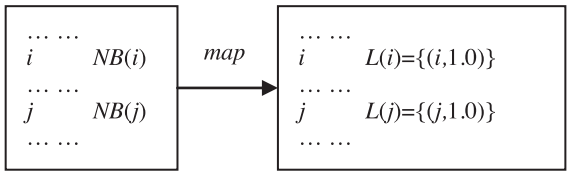


Fig. 6. Scheme of parallel vertex label initialization.

3.2.4. Parallel updating of vertex labels

The label weights should be calculated before updating the vertex labels. The neighbor list, similarity vector, and label set of each vertex i can be connected using the *join* operator to calculate label weights, as shown in Fig. 7.

For each vertex i, $\mathbf{s}_i$ and $L(i)$ are sent to each of their neighbors to calculate label weight according to Eq. (7) . The weight $w(l_k)$ for label l_k is equal to the ratio of its similarity weight $s(l_k)$ to the similarity weight of all the labels. The similarity weight of label l_k is the sum of the similarity of all the neighbors whose label sets include this label.

$$\begin{aligned} w(l_k) &= \frac{s(l_k)}{1+\sum_{p} s(l_p)} \\ s(l_k) &= \sum_{j \in LV(l_k)} s_{ij} \\ LV(l_k) &= \{i | l_k \in L(i)\} \end{aligned} \tag{7}$$

The vertex labels are updated according to the newly updated label weights, as demonstrated in Fig. 8. The parallelization process itself is simple and only requires a *map* operator. The complexity lies in the *map* operation details. For each $L'(i)$, the following steps are performed in the *map* operation.

(1) The labels in $L'(i)$ are sorted according to their weights in a descending order.
(2) The ankle value a_k of the labels is determined.
(3) The labels whose weights are smaller than ρw_{max} are dropped.
(4) The left labels form the new label set $L''(i)$.

The ankle value is used to select the position in a label weight sequence in which two consecutive weight values exhibit the sharpest fluctuation [23]. Threshold ρ is used to set up a constraint wherein labels that are too light are no longer considered [24].

3.3. The PGLPA algorithm

By combing all the parallel steps together, a parallel label propagation based on grey relational analysis, called PGLPA, is developed by combining all the parallel steps, as follows.

Algorithm 1. PGLPA

Input: network $G = (V,E)$, where V is the vertex set and E is the edge set; label selection parameter ρ; maximum iterations *maxIter*
Output: community set $\{C_i\}$

```
1. //PARALLEL ADJACENCY LIST CONSTRUCTION
2. map (key i, value j)
3.         output (key i, value j);
4.         output (key j, value i);
5. reduce (key i, values {j, k, … …})
6.         output (key i, value (NB(i));
7. //PARALLEL GREY SIMILARITY CALCULATION
8. map (key i, value NB(i))
9.         output (key partition p_k, value (i, NB(i)) );
10. mapPartition(key partition p_k, value {(i, NB(i)), (j, NB(j)), … …})
11.         output (key partition p_k, value (i, s_i));
12. reduce(key NULL, values {(i, s_i)})
13.         output (key i value s_i);
14. //PARALLEL VERTEX LABEL INITILIZATION
15. map (key i, value NB(i))
16.         output (key i, value L(i));
17. //PARALLEL LABEL PROPAGATION
18. iter ← 1
19. while iter < maxIter do
20. //PARALLEL LABEL WEIGHT CALCULATION
21.       join((key i, value NB(i)), (key i, value s_i), (key i, value L(i)));
22.             output (key i, value (NB(i), s_i, L(i)));
23.       map (key i, value (NB(i), s_i, L(i)))
24.             output (key j, value (s_i, L(i)));
25.       reduce (key i, values {(s_j, L(j)), (s_k, L(k)), … …})
26.         output (key i, value L'(i));
27. //PARALLEL VERTEX LABEL UPDATING
28.       map (key i, value L'(i))
29.         output (key i, value L''(i))
30.       if termination conditions are satisfied, then
31.         break;
32.       end if
33.    end for
34. extract communities from the vertex–label relationship;
35. output the communities;
```

3.4. Complexity analysis

The time cost for parallel adjacency list construction, parallel vertex initialization, parallel label weight calculation, and parallel vertex label updating is proportional to the number of vertices n and vertex degree k and inversely proportional to the number of machines N_m. The time cost for parallel grey similarity calculation, aside from its dependence on n and k, is also proportional to the number of partitions N_p. N_p is typically set to the number of CPU cores in a computation cluster. Thus, it can be regarded as proportional to N_m. The total time complexity of PGLPA is $O(n \times k/N_m)$, which can be further reduced to $O(m/N_m)$, where m is the number of edges.

Network data require $O(m/N_m)$ space. The space necessary for parallel computation, excluding that for parallel grey similarity calculation, is $O(n \times k/N_m)$. Parallel grey similarity calculation occupies $O(n \times k)$ because of the N_p copies of neighbor lists. Thus, the total space complexity of PGLPA is $O(n \times k)$.

4. Experiments

Comprehensive experiments were conducted on artificial and real networks to evaluate the performance of the proposed algorithm. This section presents the details of the experimental preparation and the analysis of the experimental results.

4.1. Experimental preparation

4.1.1. Datasets

The artificial networks were generated using the benchmark network generator developed by Lancichinetti and Fortunato [25]. The real social networks were obtained from the SNAP project website [26]. Table 1 summarizes the details of the experimental data sets. Parameters *n, m, mu, k, maxk, minc, maxc, on*, and *om* represent the number of vertices and network edges, the mixing parameter required by the network generator, the minimum and maximum degrees of the vertices, the average and maximum community sizes, the number of overlapping vertices, and the number of memberships of the overlapping nodes, respectively.

The sizes of the artificial networks vary from 100 k (one thousand) to 500 k (fifty thousand) for small networks and 1 M (one million) to 5 M (five million) for large networks. The ratios of the mixed vertices range from 10–80%. The degree of a vertex varies

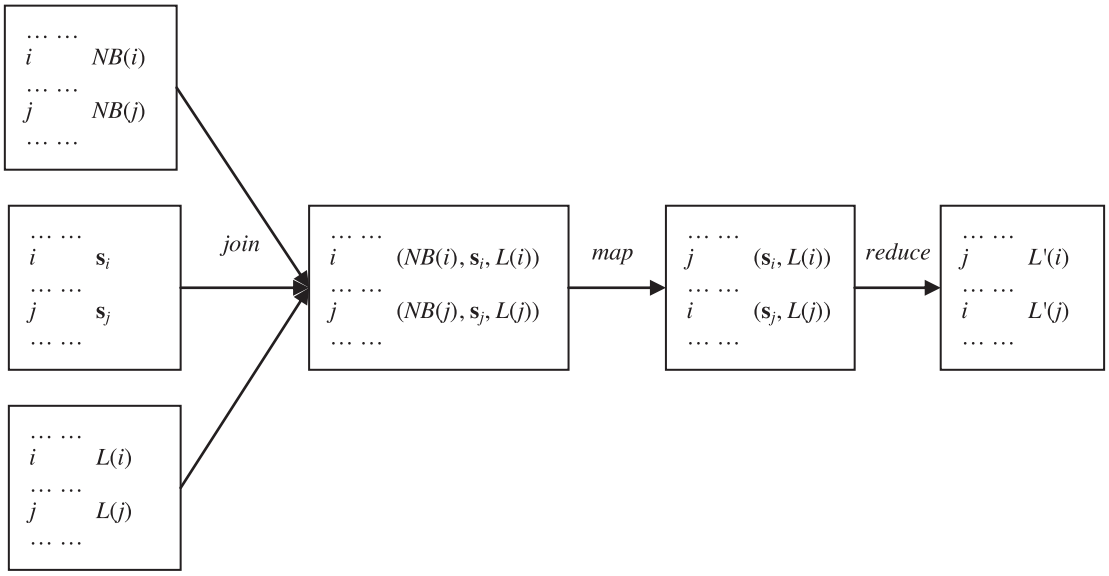


Fig. 7. Scheme of parallel label weight calculation.

Table 1
Details of the experimental data sets.

Network		Description
Artificial networks		n = 100k to 500k for small networks and 1 M to 5 M for large networks mu = 0.1∼0.8, k = 20, $maxk$ = 100, $minc$ = 10, $maxc$ = 100, on = 0.1 n, om = 3
Real networks	CA-HepPh	n = 12,008, m = 118,505
	Email-Enron	n = 36,692, m = 183,831
	web-Stanford	n = 281,903, m = 2,312,497
	com-youtube	n = 1,134,890, m = 2,987,624

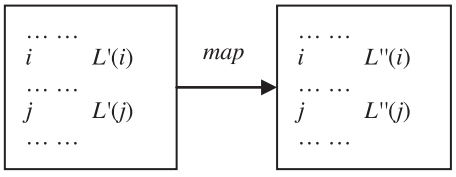


Fig. 8. Scheme of parallel vertex label updating.

from 20–100, and the size of a community varies from 50–100. The directed edges in the directed graphs are all transformed into undirected edges to obtain the corresponding undirected graphs.

4.1.2. Implementations of the parallel COPRA and SLPA

The clustering accuracy and run time of the proposed algorithm are compared with those of the COPRA and SLPA algorithms. Parallel versions of the COPRA and SLPA algorithms were developed for fair comparison. Their details are given in follows.

(1) Parallel COPRA (PCOPRA)

With modifications to the steps of PGLPA, we can design a scheme for the parallelization of COPRA. First, the parallel grey similarity calculation is no long necessary. Second, the weight of each vertex's initial label is set to 1.0. Third, in the step of parallel label weight calculation, we only need to measure the weight of a vertex's label according to its proportion in the vertex's neighbors. Finally, each vertex label is updated according to COPRA's strategy.

(2) Parallel SLPA (PSLPA)

PGLPA can also be modified to realize the parallelization of SLPA. First, the parallel grey similarity calculation is removed. Second, the initial count of each vertex's label is set to 1. Third, the parallel label weight calculation and the parallel vertex label updating are replaced with the parallel speak and listen that includes two steps: (1) parallel speak: each vertex selects a label randomly from its memory with the probability of its occurrence and sends it to its neighbors; (2) parallel listen: each vertex selects the most popular label received from its neighbors to its memory. Fig. 9 illustrates the scheme of parallel speak and listen. The *map* operator sends the selected labels l_i to vertices' neighbors. The *reduce* operator receives vertices' labels sent from their neighbors and updates their memory $L'(i)$. Finally, a postprocessing procedure is added to extract communities according to vertices' memory of labels and the threshold r.

4.1.3. Performance criteria

Clustering accuracy is measured using normalized mutual information (NMI) [27] if the community structures of a network are known or modularity Q [2] if this information is absent. The equations for NMI and Q are as follows:

$$\begin{aligned} &\mathrm{NMI}(X|Y) = 1 - [\mathrm{H}(X|Y) + \mathrm{H}(Y|X)]/2 \\ &H(X) = -\sum_{i=1}^{n} p(x_i \log x_i) \\ &H(X) = \sum_{i,j} p\left(x_i, y_j\right) \log \frac{p(y_j)}{p\left(x_i, y_j\right)} \end{aligned}, \tag{8}$$

where X and Y denote the sets of true communities and discovered communities, respectively. x_i and y_i represent a vertex in X and Y, respectively. H(X) is the entropy of X, and H($X|Y$) is the conditional

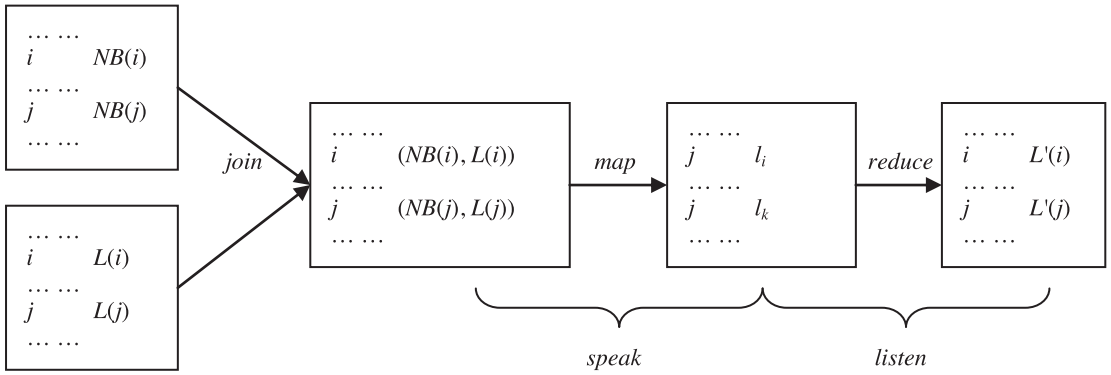
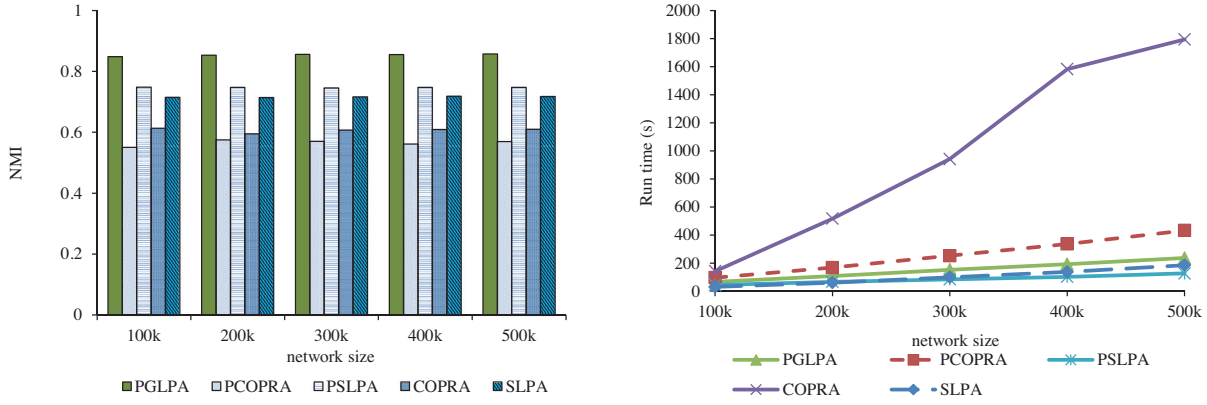


Fig. 9. Scheme of parallel speak and listen in PSLPA.



(a) NMI (b) run time

Fig. 10. NMI and run time of the algorithms on small networks with varying sizes.

entropy of X given Y.

$$Q = \sum_{c=1}^{n_c} \left[\frac{l_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right], \tag{9}$$

where m, n_c, l_c, and d_c are the edge number, the number of communities, the number of internal edges in community c, and the sum of degrees in community c, respectively.

4.1.4. Run configuration

The parameters of the algorithms are: (1) PGLPA: $\rho = 0.5$, $maxIter = 10$; (2) COPRA and PCOPRA: $v = 3$, $maxIter = 10$; (3) SLPA and PSLPA: $r = 0.5$, $maxIter = 10$. Parameters ρ and r are set to 0.5 according to the suggestion in [8]. Parameter v is set to the true number of communities (*om* in Table 1) that a vertex belongs to. The results of the experiments indicate that PGLPA converges after 10 iterations. Hence, the *maxIters* of all the algorithms are set to 10 for fair comparison.

PGLPA is implemented on the Spark framework [28]. All the experiments are performed on a cluster composed of eight machines virtualized using VSphere 5.5. All the machines have the same configuration, as follows: 2.0 GHz 2-core CPU, 8GB RAM, and CentOS 6.5.

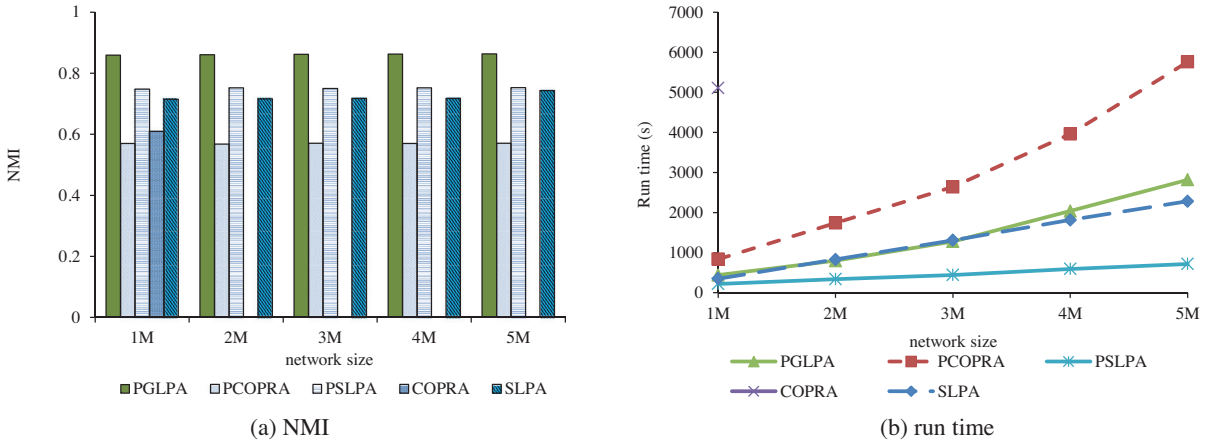
4.2. Experimental results for the artificial networks

(1) Effect of network size

The experimental results for small networks with varying sizes are presented in Fig. 10.

As shown in Fig. 10(a), the clustering accuracy of the PGLPA algorithm is consistently better than the other algorithms on small networks. This result demonstrates the effectiveness of grey similarity and automatic label selection through the ankle values. Moreover, constraint on minimum label weight also helps stabilize identified communities. It is interesting to find that the performance of COPRA and PCOPRA is not identical and neither is the performance of SLPA and PSLPA. It may be caused by the synchronous label updating strategy used in PCOPRA and PSLPA. The synchronization mechanism is necessary for designing the parallel steps of the algorithms. On the contrary, COPRA and SLPA update the labels asynchronously. Therefore, the updated labels of the early processed vertices can affect the vertices processed later. Fig. 10(b) shows that the time cost of all the algorithms increases nearly linearly with network size except for COPRA. When network size is larger than 300 k, the runtime of COPRA increases rapidly due to the memory constraint. For small networks, the time spent on data processing (work time) is comparable with the time spent on cluster administration and communication (extra time). Hence, the speedup of the other algorithms caused by parallel computation is not evident. It is possible that the parallel algorithms run slower than the single-machine algorithms when the extra time far exceed the work time. Moreover, the time cost of calculating Jaccard similarity and grey similarity, as well as that of searching for ankle values, hinders PGLPA. However, the experimental results for large networks demonstrate that the power of parallel computation becomes remarkable when network scale increases beyond the capability of a single-machine algorithm such as COPRA.

Please cite this article as: Q. Zhang et al., A social community detection algorithm based on parallel grey label propagation, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.06.002



(a) NMI (b) run time

Fig. 11. NMI and run time of the algorithms on large networks with varying sizes.

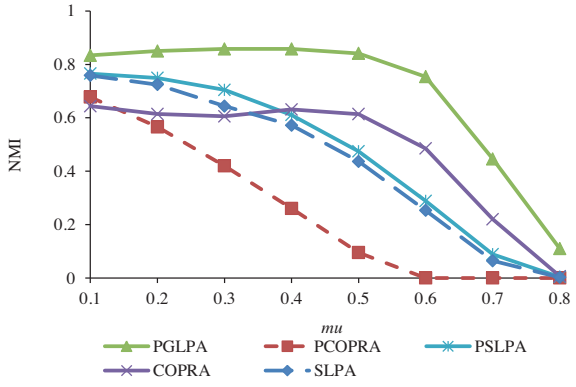


Fig. 12. NMI of the algorithms with varying *mu* values.

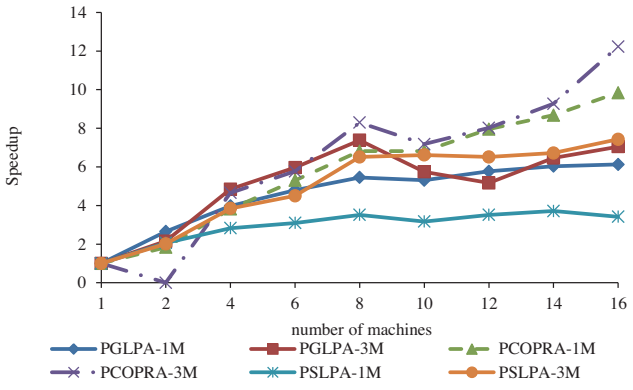


Fig. 13. Speedup of the algorithms.

The experimental results for large networks with varying sizes are presented in Fig. 11. The results of COPRA are absent on the networks of 2 M to 5 M because it runs out of memory on the networks.

Fig. 11(a) shows that the clustering accuracy of all the algorithms on large networks are nearly the same as those on small networks. PGLPA is again consistently better than the other algorithms. From Fig. 11(b), we can see that the run time of PGLPA increases nearly linearly with network size, which reflects the good scalability of the algorithm. PCOPRA algorithm also scales well with the increase in network size. However, its run time is always higher than that of PGLPA. COPRA can only run on the 1 M network and its time cost is extremely high. SLPA and PSLPA run faster than PGLPA and PCOPRA because the speak and listen strategy is simpler than the label updating strategies used in PGLPA. PSLPA exhibits better scalability than SLPA, which is largely due to the parallel speak and listen scheme developed in Fig. 9. In the real world, as a smoothly scalable algorithm is important for efficiently analyzing large networks because many social networks contain millions or even billions of vertices.

(2) Effect of mixing degree

The experimental results for the 100k network with varying *mu* values are presented in Fig. 12.

Fig. 12 shows that the clustering accuracy of all the algorithms except COPRA is stable when the value of *mu* is smaller than 0.5. Subsequently, clustering accuracy decreases as the value of *mu* increases. The poor performance of PCOPRA shows that the synchronous label updating used for parallelizing COPRA has great impact on the clustering accuracy. The performance of PGLPA is consistently better than those of the other algorithms at all *mu* values. Automatic label selection through the ankle values and constraint on the minimum label weight help strengthen the stability of the vertex labels. Therefore, the effect of monster communities on the vast number of nearby small communities is suppressed.

(3) Effect of cluster scale

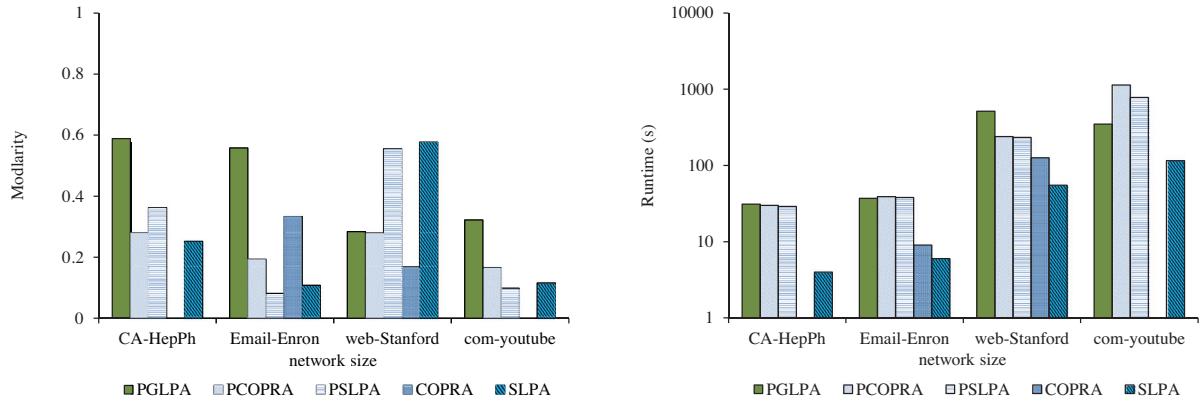
Experiments are conducted on the 1 M network with a varying number of machines to evaluate the effect of cluster scale on the performance of PGLPA, PCOPRA and PSLPA. The speedup is calculated using Eq. (10) :

$$Speedup = \frac{\text{run time on the single machine}}{\text{run time on the cluster}}. \quad (10)$$

The experimental results for the 1 M and 3 M networks with a varying number of machines are presented in Fig. 13.

Fig. 13 shows that a boost on running speed caused by adding machines to the cluster is evident when the number of machines is smaller than 8. Subsequently, the advantage of expanding the cluster begins to decrease. At the beginning, communication and management costs among the machines in a moderate computation cluster scale are negligible. Therefore, running speed increases steadily when the CPU and memory resources enrolled in the cluster are increased. However, communication and management costs become sufficiently high when the cluster expands to a certain degree, which cancels the advantage of increasing computation power. Hence, speedup slows down when the number of machines increases across a certain point. The speedup of all the algorithms on the 3 M networks is consistently better that on the 1 M networks, which means that the parallel computation model is particularly suitable for large networks. The speedup of PCOPRA

Please cite this article as: Q. Zhang et al., A social community detection algorithm based on parallel grey label propagation, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.06.002



(a) NMI (b) run time

Fig. 14. Modularity and run time of the algorithms on the real networks.

is the most remarkable since its parallel scheme is the simplest. The speedup of PGLPA is similar to PSLPA on the 3 M networks and is better than PSLPA on the 1 M networks. Additional similarity calculation steps and complex label updating strategies hinders PGLPA. PCOPRA runs out of memory on the 3 M network when the number of machines is reduced to two. Therefore, the speedup value of the PCOPRA-3 M curve at the point of two machines is zero.

4.3. Experimental results for the real networks

The experimental results for the real networks are presented in Fig. 14.

As shown in Fig. 14(a), the performances of the algorithms on the real networks are considerably different from those on the artificial networks. First, real networks generally do not present obvious community components. Therefore, clustering accuracy on real networks is not as high as that on artificial networks. Second, real networks may contain complex community structures that make community detection a time-consuming task. Therefore, the vertical axis scale of Fig. 14(b) is set to the power of 10. On nearly all the real networks, PGLPA is superior to the other algorithms in terms of both clustering accuracy and run time. This finding indicates that grey similarity calculation, automatic label selection through ankle values, and constraint on the minimum label weight also affect real networks. COPRA fails to run on the CA-HepPh network and the com-youtube network. Therefore, the results of COPRA on these two networks are empty in Fig. 14.

5. Conclusions

LPA, an efficient community detection algorithm, exhibits both merits and defects. In this study, we investigate the parallelization of the procedures of LPA and propose a fully parallel LPA. The calculation of vertex similarity on which label updating is based is parallel. Grey relational degree is integrated into Jaccard similarity to provide information for measuring similarity. The initialization and updating of the labels are also run in parallel, and multiple labels are allowed to be assigned to a vertex. Therefore, the proposed algorithm can discover overlapping communities. The experiments conducted on artificial and real networks demonstrate that the proposed algorithm runs well on both small and large networks.

Several issues remain for further research. First, the label updating strategy can be enhanced to consider label weighting manners. Second, the algorithm can be improved further to be practical by considering both overlapping and hierarchical communities. Third, with the rapid emergence of new parallel computation frameworks aside from Hadoop and Spark, efficient parallel operators may be utilized to improve the performance of the algorithm.

Acknowledgments

This work is partly supported by the National Natural Science Foundation of China under Grant no. 61300104, the Fujian Province High School Science Fund for Distinguished Young Scholars under Grant no. JA12016, the Program for New Century Excellent Talents in Fujian Province University under Grant no. JA13021, the Fujian Natural Science Funds for Distinguished Young Scholar under Grant no. 2014J06017, the Technology Innovation Platform Project of Fujian Province under Grants nos. 2009J1007 and 2014H2005, the Fujian Collaborative Innovation Center for Big Data Applications in Governments, and the Natural Science Foundation of Fujian Province under Grant nos. 2013J01230 and 2014J01232.

References

[1] Facebook.
[2] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (2010) 75–174.
[3] B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, Bell. Syst. Tech. J. 49 (1970) 291–307.
[4] P. Symeonidis, N. Iakovidou, N. Mantas, Y. Manolopoulos, From biological to social networks: Link prediction based on multi-way spectral clustering, Data Knowl. Eng. 87 (2013) 226–242.
[5] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. U.S.A. 99 (2002) 7821–7826.
[6] U. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Phys. Rev. E. 76 (2007) 12.
[7] S. Gregory, Finding overlapping communities in networks by label propagation, New J. Phys. 12 (2010) 103018.
[8] J. Xie, B.K. Szymanski, Towards linear time overlapping community detection in social networks, in: 16th Pacific-Asia Conf. Knowl. Discov. Data Min., Springer, Kuala Lumpur, Malaysia, 2012, pp. 25–36.
[9] 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.
[10] MPI Forum, The Message Passing Interface (MPI) standard (2013) http://www.mcs.anl.gov/research/projects/mpi/standard.html.
[11] K. Kuzmin, S.Y. Shah, B.K. Szymanski, Parallel overlapping community detection with SLPA, in: Proc. – Soc. 2013, 2013, pp. 204–212.
[12] W. Zhao, V. Martha, X. Xu, PSCAN: a parallel structural clustering algorithm for big networks in mapreduce, in: 2013 IEEE 27th Int. Conf. Adv. Inf. Netw. Appl., IEEE Computer Society, Barcelona, Spain, 2013, pp. 862–869.
[13] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[14] H. Lou, S. Li, Y. Zhao, Detecting community structure using label propagation with weighted coherent neighborhood propinquity, Phys. A Stat. Mech. Its Appl. 392 (2013) 3095–3105.

[15] Z. Lin, X. Zheng, N. Xin, D. Chen, CK-LPA: efficient community detection algorithm based on label propagation with community kernel, Phys. A Stat. Mech. Its Appl. 416 (2014) 386–399.
[16] S. Li, H. Lou, W. Jiang, J. Tang, Detecting community structure via synchronous label propagation, Neurocomputing 151 (2015) 1063–1075.
[17] K. Guo, Q.S. Zhang, Detecting communities in social networks by local affinity propagation with grey relational analysis, Grey Syst.: T&A 5 (1) (2015) 31–40.
[18] K. Guo, W.Z. Guo, Y.Z. Chen, Q.R. Qiu, Q.S. Zhang, Community discovery by propagating local and global information based on the MapReduce model, Inf. Sci. 323 (2015) 73–93.
[19] R.R. Li, W.Z. Guo, K. Guo, Q.R. Qiu, Parallel multi-label propagation for overlapping community detection in large-scale networks, MIWAI (2015) 351–362.
[20] S.F. Liu, N.M. Xie, The Grey System Theory and its Applications, Science press, Beijing, China, 2008.
[21] Q.S. Zhang, Difference Information Theory of Grey Hazy Set, Petroleum Industry Press, Beijing, China, 2002.
[22] X. Zhang, J. Zhu, Q. Wang, H. Zhao, Identifying influential nodes in complex networks with community structure, Knowl.-Based Syst 42 (2013) 74–84.
[23] J.B. Huang, H.L. Sun, D. Bortner, Y.G. Liu, Mining hierarchical community structure within networks from density-connected traveling orders, J. Softw. 22 (5) (2011) 951–961.
[24] 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.
[25] A. Lancichinetti, S. Fortunato, Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities, Phys. Rev. E. 80 (2009) 1–8.
[26] J. McAuley, J. Leskovec, Learning to discover social circles in ego networks, in: Proc. 26th Annu. Conf. Neural Inf. Process. Syst. 2012, Lake Tahoe, USA, 2012, pp. 548–556.
[27] A. Lancichinetti, S. Fortunato, J. Kertész, Detecting the overlapping and hierarchical community structure in complex networks, New J. Phys. 11 (2009) 033015.
[28] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M.J. Franklin, S. Shenker, I. Stoica, Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing, in: Proc. of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI'12), 2012, p. 2.

Please cite this article as: Q. Zhang et al., A social community detection algorithm based on parallel grey label propagation, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.06.002

Qishan Zhan received the B.S. degree in exploration from College of Daqing Petroleum Institute, Heilongjiang, China, in 1985, the M.S. degree in Management System Engineering from Harbin Institute of Technology, Harbin, China, in 1993, and the Ph.D. degree in Automatic Control from Huazhong University of Science and Technology, Wuhan, China, in 1996. His research interests include grey system, business intelligence, physical distribution management and system engineering.

Qirong Qiu received the M.S. degrees in Computer Science from Fuzhou University, Fuzhou, China, in 2010. He is now a Ph.D. Student of management system engineering in Fuzhou University. His research interests include the grey system, data mining on big data and cloud computing.

Wenzhong Guo received the B.S. and M.S. degrees in computer science, and the Ph.D. degree in communication and information system from Fuzhou University, Fuzhou, China, in 2000, 2003, and 2010, respectively. He is currently a full professor with the College of Mathematics and Computer Science at Fuzhou University. His research interests include intelligent information processing, sensor networks, network computing, and network performance evaluation. Currently, he leads the Network Computing & Intelligent Information Processing Lab, which is a key Lab of Fujian Province, China. He is a member of ACM, a senior member of China Computer Federation (CCF).

Kun Guo received the B.S. and M.S. degrees in Computer Science from Fuzhou University, Fuzhou, China, in 2002 and 2005, respectively, and the Ph.D. degree in management system engineering from Fuzhou University in 2012. He is currently an associate professor with the College of Mathematics and Computer Science at Fuzhou University. His research interests include the grey system, data mining on big data and cloud computing. He is a member of ACM and a member of China Computer Federation (CCF).

Naixue Xiong is currently a professor with the Department of Business and Computer Science at Southwestern Oklahoma State University. His research interests include Cloud Computing, Virtual Computing, Fault Tolerance, Distributed Systems/Networks and Artificial Intelligence.

Please cite this article as: Q. Zhang et al., A social community detection algorithm based on parallel grey label propagation, Computer Networks (2016), http://dx.doi.org/10.1016/j.comnet.2016.06.002