Detecting community structure using label propagation with consensus weight in complex network*

Liang Zong-Wen(梁宗文)[a)b)†], Li Jian-Ping(李建平)[a)], Yang Fan(杨 帆)[a)], and Athina Petropulu[b)]

[a)] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[b)] Electrical and Computer Engineering Department, Rutgers, The State University of New Jersey, NJ 08854, USA

(Received 13 January 2014; revised manuscript received 26 March 2014; published online 16 July 2014)

Community detection is a fundamental work to analyse the structural and functional properties of complex networks. The label propagation algorithm (LPA) is a near linear time algorithm to find a good community structure. Despite various subsequent advances, an important issue of this algorithm has not yet been properly addressed. Random update orders within the algorithm severely hamper the stability of the identified community structure. In this paper, we executed the basic label propagation algorithm on networks multiple times, to obtain a set of consensus partitions. Based on these consensus partitions, we created a consensus weighted graph. In this consensus weighted graph, the weight value of the edge was the proportion value that the number of node pairs allocated in the same cluster was divided by the total number of partitions. Then, we introduced consensus weight to indicate the direction of label propagation. In label update steps, by computing the mixing value of consensus weight and label frequency, a node adopted the label which has the maximum mixing value instead of the most frequent one. For extending to different networks, we introduced a proportion parameter to adjust the proportion of consensus weight and label frequency in computing mixing value. Finally, we proposed an approach named the label propagation algorithm with consensus weight (LPAcw), and the experimental results showed that the LPAcw could enhance considerably both the stability and the accuracy of community partitions.

Keywords: label propagation algorithm, community detection, consensus cluster, complex network

PACS: 89.75.Fb, 89.75.Hc **DOI:** 10.1088/1674-1056/23/9/098902

1. Introduction

In real society, many systems can be described as networks, with vertices representing the presence of entities and edges signifying some sort of relationship. These networks are highly complex and exist in various areas, such as networks of web pages, social networks, biological networks, and information networks. The study of network structure has attracted significant attention, especially in the field of detecting communities of networks. Communities can be deemed as groups of vertices which share common properties or play similar roles within the network.[1,2] In these networks, relationships between vertices within the same community are dense, while relationships are sparser between vertices from different communities. The general goal of community detection is to allocate meaningful divisions into groups by studying structures of the whole network. During the last decades, many methods have been proposed to find the communities contained in a network, see Refs. [3] and [4] for a survey. Hence, many improvements with some classic algorithms have also been achieved. Zou *et al.*[5] proposed a community detection algorithm with consideration of non-topological information, which argued that in some real-world networks, some unknown non-topological information dominate community division. Tang *et al.*[6] proposed optimization-based method of identifying the topological structure of a complex network, which can identify not only the non-synchronization complex networks, but also the synchronization complex networks. Wang *et al.*[7] improved the fast uncovering community algorithm designed by Blondel *et al.*,[8] and it has shown these improvements can enhance the performance of community detection greatly.

However, most of these methods are based on the maximization of a quality function known as modularity which is often used to measure the internal density of communities.[9] Some researchers have proved that modularity has many limitations. For example, it is a NP-complete problem for modularity maximization.[10] Further studies show that the modularity may lead to determining the resolution limit in identifying a small group from large networks.[11] Furthermore, as using heuristics, the modularity could lead to inaccurate results.[12] For example, in random networks, these algorithms may discover communities with a high modularity but no any community structure exists in these networks.[12] Moreover, most of these methods are limited in very large-scale networks because of high time complex or no *priori* knowledge.[13]

A notable method was proposed by Raghavan *et al.*,[14] who employed a simple label propagation algorithm (LPA) to reveal communities in networks. This innovative and promising algorithm only uses the network structure as a guide. Due to extremely fast structural inference of label propagation, the

*Project supported by the National Natural Science Foundation of China (Grant No. 61370073) and the China Scholarship Council, China (Grant No. 201306070037).

†Corresponding author. E-mail: zongwen-liang@hotmail.com

© 2014 Chinese Physical Society and IOP Publishing Ltd

algorithm exhibits near linear time complexity. Therefore, label propagation algorithm is a good choice for large-scale networks with millions of nodes and edges.[15]

Nevertheless, the randomness issue of label propagation has not yet been properly addressed. As randomness is introduced into LPA, different communities may be detected in different runs over the same network.[16] In this paper, in order to improve the stability of community results, we employed a consensus cluster to find consensus weighted graph. In this graph, the weight of the edge was the proportion of the times number which two nodes of the edge were allocated in the same community over the total consensus times number. By combining consensus weight and label frequency as a mixing value, we proposed a community detection algorithm whereby the label of a node was updated to the label with the maximum mixing value. We also introduced a proportion parameter to coordinate the role of consensus weight and label frequency in label updating steps. The results showed that our algorithm exhibited better performance than original LPA in which either synchronous or asynchronous updating rules be adopted.

The remainder of the paper is organized as follows: in Section 2, we introduce the label propagation algorithm and in Section 3, consensus clustering is reviewed. Our algorithm (LPAcw) is described in Section 4. The experimental results are represented in Section 5, followed by our conclusion in the last section.

2. Label propagation algorithm

In this paper, we just discuss the unweighed and undirected networks, and we do not consider isolated nodes in networks. In LPA, each node in a network is given a unique label initially, indicating the community it belongs to. At each propagation step, each node updates its current label by choosing the label which the most of its neighbors have (the maximal label). Formally, the label updating rule of node v is

$$l_v^{\text{new}} = \arg\max_l |N^l(v)|, \tag{1}$$

where l_v^{new} is the new label of node v, $N^l(v)$ is the set of node v's neighbors that have the label l. When there are multiple maximal labels among the neighbors' labels, the new label is picked randomly from them. The label propagation step is performed iteratively until all nodes have a label that is (one of) the most frequent label (labels) of its neighbors. Communities are identified as a group of nodes sharing the same label.

The biggest advantage of LPA is that it has a less expensive computation cost than what has been possible so far, and its time complexity has been proved to near linear time complexity $O(km)$, where k is the number of propagation iterations and m is the number of edges. It has been proved that 95% of nodes can be identified correctly after five iterations.[14]

However, when this algorithm was proposed in Ref. [14], Raghavan *et al.* indicated that LPA is not stable: as arranging the nodes in a random order in each iteration step, different communities may be identified in different runs over the same network. The stability of LPA is a main issue and many improvements have been done in this field. Most of them employed a control factor (or a set of factors) and incorporated it into the update rule. Šubelj *et al.*[16] employed a balanced propagation to counteract the randomness introduced by random updates. Xie *et al.*[17] introduced a set of operators to control and stabilize the propagation dynamics, which compared with LPA it can significantly improve the quality of detected communities. A good method has been proposed recently by Lou *et al.*,[13] they found that sharing only one-hop label in LPA may lead to much network topology information being lost, hence they introduced a measure named weighted coherent neighborhood propinquity (weighted-CNP) to represent the probability that a pair of vertices were involved in the same community. They integrated weighted-CNP to LPA and proposed the LPA-CNP algorithm to detect community structure, which was more robust and effective than LPA in large-scale networks.

These improvements mentioned above show good performance and strong stability in community detection. However, if we review the design concept of LPA carefully, we can find that the instability is inevitable in LPA. On the one hand, in the label updating step, if there are multiple labels shared by most nodes among neighbors, one label is picked randomly. On the other hand, the order of node updates within asynchronous label propagation governs the algorithm's dynamics in a similar manner as the corresponding node propagation preferences.[16] Hence, how to enhance the stability of LPA is still an open problem in both synchronous and asynchronous models.

3. Consensus clustering

Consensus clustering is a well-known technique for data analysis to generate stable results out of a set of partitions delivered by stochastic methods.[18–20] By consensus clustering, some consensus partitions are generated. Lancichinetti *et al.*[21] applied a non-deterministic community detection algorithm on a network many times and got a consensus matrix. In this consensus matrix, each element indicated how many times the vertex pairs were allocated in the same community during all iterations. By further splitting the consensus matrix using the same non-deterministic community detection algorithm constantly, the result was a block 0–1 diagonal matrix, which delivered the community structure of the original network. In order to make the result more accurate, Seifi *et al.*[2] introduced a threshold $\alpha \in [0, 1]$ to reduce noise, and removed elements in the consensus matrix where the value was lower than α. Both of their works can be summarized in the following three steps:

1) Apply a non-deterministic community detection algorithm enough times on a network graph.

2) Compute the consensus matrix of which each element represents the times of node pairs belonging to the same community.

3) Further split the consensus matrix using the same non-deterministic community detection algorithm, until the consensus matrix becomes a block 0–1 diagonal matrix.

Consensus clustering provides a method to represent the consensus across multiple runs of a clustering algorithm and to assess the stability of the discovered clusters. It can also be used to represent the consensus over multiple runs of a clustering algorithm with a random parameter.[17] In a random scene, finding a consensus solution by consensus clustering to improve the stability of LPA is a promising choice in this field.

4. Label propagation with consensus weighted graph

4.1. Consensus partition

Consensus partitions refer to a set of community detection results generated by repeatedly executing the same community partition algorithms on a network. It has been mentioned above that the instability of LPA is mainly embodied in that different runs can generate different communities in the same network. Obviously, a network graph split by LPA often has more than one result of different community detection. A better option is to combine all the various results of community detection into a new partition, and to get a more stable partition result.

We introduce the consensus clustering technique to get consensus partitions by applying the basic label propagation algorithm on a network graph multiple times, and each consensus partition should optimize

$$f = \max\left(\sum_{i=1}^{k}(|u \in C_{v^i}|) - \sum_{i=1}^{k}(|u \notin C_{v^i}|)\right), \tag{2}$$

where $N^i(v)$ is the set of neighbors of node v that have label i and C_{v^i} is the label i that node v has. The k is the number of the label node v has, and $u \in N^i(v)$.

Viewing this constraint from the set, equation (2) can be rewritten as follows:

$$f = \max\left(\sum_{i=1}^{k}(|N^i(v) \cap C_{v^i}|) - \sum_{i=1}^{k}(|N^i(v) \cap (1 - C_{v^i}|))\right). \tag{3}$$

Equation (3) is a good description of each consensus partition, which restricts that node v and most of its neighbor nodes with the same label should be allocated in the same partition.

4.2. Consensus weighted graph

As discussed above, by multiplying the basic label propagation algorithm on a network, it generates a set of community partitions which can be used to compute the number of partitions that two nodes are assigned in the same cluster. We divide this number by the total partitions number for normalization and note it as the consensus weight. We calculate the consensus weight by

$$C_{\text{weight}(i,j)} = \frac{\sum_{k=1}^{n_p}(P_{ij}^k)}{n_p}, \tag{4}$$

where $P_{ij}^k = 1$, if node i and node j are assigned to the same cluster in the k-th iteration, otherwise 0, and n_p is the total number of iterations.

We calculate the consensus weight for all nodes to obtain a consensus weighted matrix. Based on this matrix, we can create a consensus weighted graph. In consensus weighted graphs, the bigger consensus weights exist in edges where node pairs of the edge have the most frequent co-clusters, whereas lower consensus weights indicate that node pairs corresponding to these edges are probably at the boundary between different partitions. To maintain the large consensus weights and to drop the low ones, we introduce a threshold $\alpha \in [0,1]$ to filter noise. As Fig. 1 shows, weights in edges which between red nodes and blue nodes are 0 when α is set to 0.3.

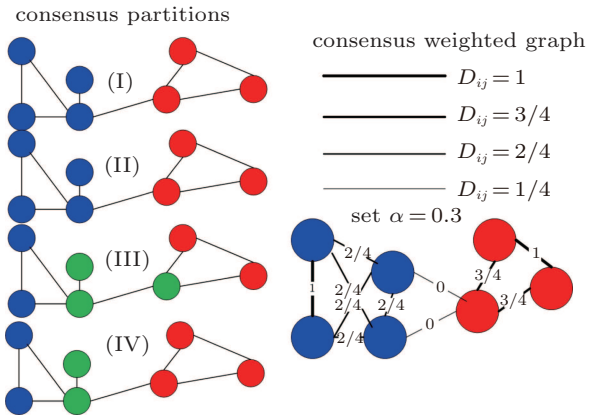


Fig. 1. (color online) Consensus weighted graph.

The essence of a consensus weighted graph is to give a preference to each node, and the preference can specify which community the node more likely belongs to. Higher preference is given to node pairs with more times allocated in the same cluster during all partitions, and vice versa.

We rewrite Eq. (3) with consensus weighted form as follows:

$$f = \max\left(\sum_{i=1}^{k}(C_{\text{weight}}(|N^i(v) \cap C_{v^i}|))\right)$$

$$-\sum_{i=1}^{k}(C_{\text{weight}}(|N^i(v)\cap(1-C_{v^i}|)))\Big), \tag{5}$$

where k is the number of labels of node v, and $C_{\text{weight}}(X)$ represents the consensus weight of edges in set X.

4.3. Label propagation with consensus weight

In the basic label propagation algorithm, the label of each node is updated to the label shared by most of its neighbors. Šubelj *et al.*[15] first analyzed how different node orders can affect the final outcome of the algorithm. They studied the label propagation algorithm on a toy sample network as Fig. 2(a). The network consisted of two communities, namely c_1 and c_2, which were defined in a strong sense[23] (i.e., each node has more intra-community than inter-community edges). Their labels were represented by the colors of the nodes.

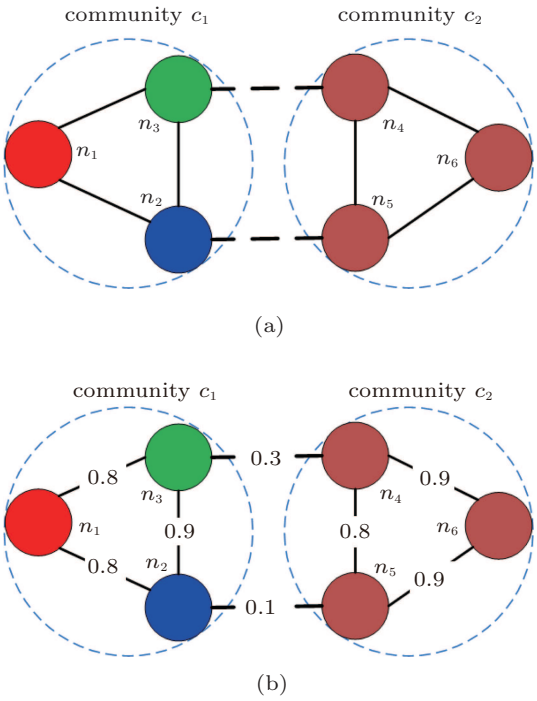


Fig. 2. (color online) Toy sample networks. (a) The unweighted graph. (b) The weighted graph.

When node n_1 is considered first to update its label in label propagation process, it will pick a label from nodes n_2 or n_3 randomly. Assuming that node n_2 is chosen first, node n_1 becomes blue. Whether node n_2 or n_3 is updated next, at the end of this iteration, all nodes in community c_1 will be updated with the same label (blue), and the outcome is the natural community structure of the network. On the contrary, if node n_1 is updated last, it can get a different result. Again, we assume that node n_2 is updated before n_3. If unfortunately node n_2 updates its label with node n_5, it is straightforward to see that gradually nodes in community c_1 may all update to the label with node n_5, and at the end of this iteration, all nodes in the network will be classified as the same community c_2. However, if we select the label according to the consensus weight in the edges between node n_2 and its neighbors (see Fig. 2(b)), node n_5 will certainly be abandoned when updating node n_2's label because the consensus weight is $C_{\text{weight}(2,5)} < C_{\text{weight}(2,3)}$. Therefore, adopting the consensus weight as the preference to order the neighbor nodes in an LPA can improve the quality of the LPA.

As mentioned in Section 2, random node updates cannot be avoided. Therefore, the consensus weight can seem to be an indicator of selecting a label, and it can be used to counteract the randomness of the LPA. We introduce the consensus weight into the LPA and call this method LPAcw. In order to generalize, we also use a proportion parameter λ to control the proportion of consensus weight and label frequency in label updating steps.

We rewrite the update rule of LPA in LPAcw below:

$$l_v^{\text{new}} = \arg\max_l(\lambda|N^l(v)| + (1-\lambda)(C_{\text{weight}}(v,u))), \quad \forall u \in N^l(v), \tag{6}$$

where $N^l(v)$ is the set of nodes that have the label l in v's neighbors and $C_{\text{weight}}(v,u)$ is the consensus weight of the edge between nodes v and u. The $\lambda \in [0,1]$ is a proportion parameter which has different values in different networks. If $\lambda = 0$, the updating rule is completely determined by consensus weights of edges between node v and its neighbors, while if $\lambda = 1$, then the LPAcw is just the basic definition of the LPA.

The steps of LPAcw are as follows:

i) Allocate a distinct label for each node initially.

ii) Apply a basic label propagation algorithm n_p times to a network graph with the initial label and get n_p community partitions. The updating rule of the LPA can be synchronous or asynchronous. Each partition is constrained by Eq. (3).

iii) Calculate the consensus weight for each node pairs according to Eq. (4) and create the consensus weighted graph of the original network.

iv) Arrange all nodes in the original network in a random order and set it to be X.

v) For each $x \in X$ chosen in random order, update its label with the label which has the maximum mixing value according to Eq. (6) among its neighbors and ties are broken uniformly randomly.

vi) If every node has a label which has the maximum mixing value between itself and neighbors, then stop the algorithm. Else, go back to step iv).

As the consensus weight is calculated by executing the basic label propagation algorithm on the network multiple times, if the iteration number is selected properly, we can get an optimal solution of node preference. Wang *et al.*[24] indicated that it is possible to terminate the iteration when the variation of the consensus weight is small enough. Similarly, threshold α has a strong influence on the results of calculating the consensus weight. However, in this paper, to simplify

the problem, we derive no theoretical bound on the minimum number of executions and threshold α, and analyze the results under different values.

4.4. Complexity analysis

Let the network be represented by a simple undirected graph $G = (N, E)$, where N is the set of nodes and E is the set of edges, $n = |N|$ is the number of nodes, and $m = |E|$ is the number of edges.

Firstly, in consensus clustering steps, each execution of label propagation requires a time of $O(km)$ (k is the number of propagation iterations),[14] and all executions need $O(n_p(km))$ time, where n_p is the number of executions. Then, in constructing consensus weighted graph steps, calculating weights for all edges need $O(m)$ time. Finally, in LPAcw steps, the time complexity is the same as the LPA of $O(km)$. Hence, the overall time complexity is $O(n_p(km)) + O(m) + O(km)$.

Therefore, LPAcw keeps the advantage of high speed of the LPA, with near linear time.

5. Experiments and results

5.1. Different measures of community structure significance

The most popular measure of the significance of communities is modularity Q,[9] which measures the significance of communities due to some null models and is often expressed as

$$Q = \sum_{i=1}^{N_c} \left[\frac{I_i}{m} - \left(\frac{D_i}{2m} \right)^2 \right], \tag{7}$$

where I_i is the number of intra-community edges that have both ends in community i, D_i is the sum of all degrees of nodes in community i and N_c is the total number of communities.

Another commonly adopted measure is normalized mutual information (NMI),[25] which is a *de facto* standard in community detection literature. NMI often is defined as

$$\mathrm{NMI}(A,B) = \frac{2I(A,B)}{H(A)+H(B)}, \tag{8}$$

where A and B can correspond randomly to the partitions of a network extracted by a community detection algorithm and the real partitions for this network respectively, $I(A,B)$ represents the mutual information between partitions A and B, $H(A)$ and $H(B)$ are standard and conditional entropies. If the partitions detected by an algorithm can identify the real partitions, then $\mathrm{NMI}(A,B) = 1$ and $\mathrm{NMI}(A,B) = 0$ for the found partitions are totally independent of the real partitions.

5.2. LFR benchmark network

We have first analyzed the performance of LPAcw on a class of synthetic benchmark networks with planted partition.[26] In this network, there is a mixing parameter $\mu \in [0,1]$ which is a measure of the degree of fuzziness of community structure. If μ has a low value, it means there

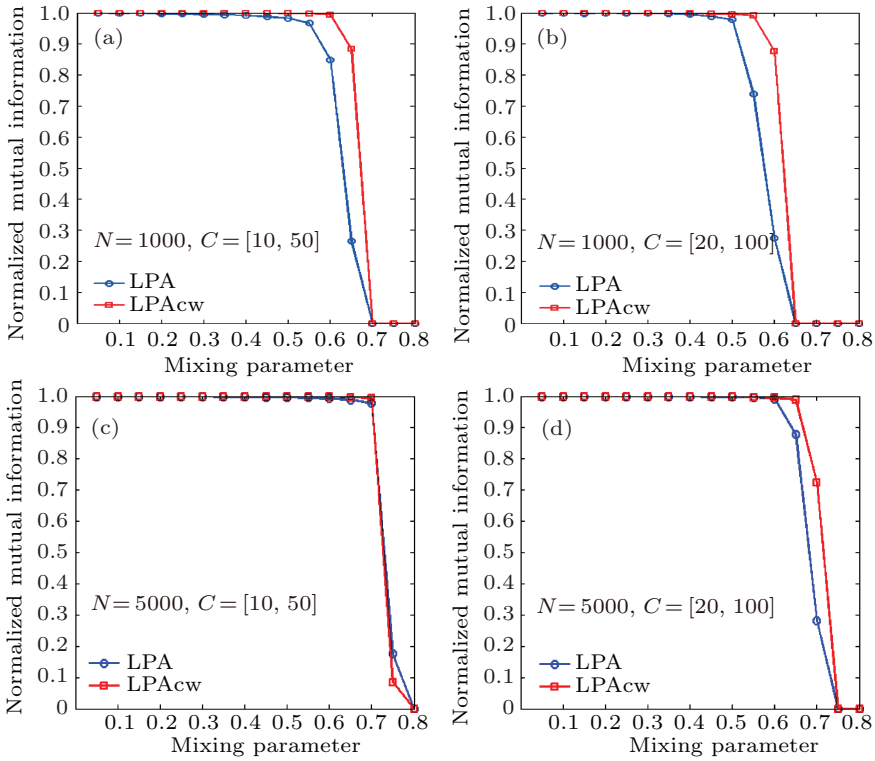


Fig. 3. (color online) LPAcw on the LFR benchmark. The parameters of the LFR benchmark are: average degree $k = 20$, maximum degree $k_{\max} = 50$, minimum community size $c_{\min} = 10$, maximum community size $c_{\max} = 50$, the degree exponent is $t_1 = 2$, the community size exponent is $t_2 = 3$. (a) $N = 1000$, $C = [10,50]$; (b) $N = 1000$, $C = [20,100]$; (c) $N = 5000$, $C = [10,50]$; (d) $N = 5000$, $C = [20,100]$.

are clear community structures in the network. In Fig. 3, the benchmark graphs consist of 1000 and 5000 vertices, respectively. We set the iteration number of consensus clustering as 200, the threshold $\alpha = 0.5$, and proportion parameter $\lambda = 0.6$. Although a mixing parameter μ was set, from Fig. 3, we can see that partitions from LPA are significantly different from the planted partition of the benchmark, whereas LPAcw can get better partitions than LPA because of consensus clusters. The results confirmed that the LPAcw shows much better performance than the LPA.

5.3. Real network

We apply the LPAcw on some common real networks where the community structures are well known; we will analyze the performance of the LPAcw from stability, modularity and different proportion parameter λ.

Stability LPAcw can obtain stable partitions as the consensus clusters are integrated to the LPA. We increase the run number n_r, and compare the LPAcw with the LPA by executing these two algorithms on networks multiple times. Figures 4 and 5 show the comparison results.

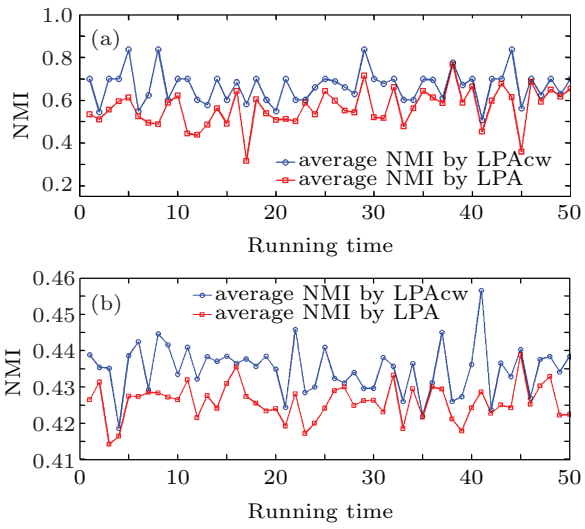


Fig. 4. (color online) The average NMI comparison of LPA and LPAcw in Karate (a) and Dolphins (b).

On the one hand, as the consensus weight calculated by executing the basic LPA multiple times, it is proper to compare the NMI of consensus clusters obtained by the LPA and the NMI of partitions obtained by the LPAcw. In the LPAcw, it works by using the former consensus clusters in the same iteration. Here we first use the LPA to obtain 50 consensus clusters of networks, then run the LPAcw 50 times on the same network by using the former 50 consensus clusters, and finally we get 50 average NMI of LPAcw. We compare these 50 NMIs of LPAcw and 50 NMIs of consensus clusters. Figure 4 shows this comparison in Karate and Dolphins. From this comparison, we hope to verify whether the LPAcw can improve the NMI of partitions compared with the LPA, so this comparison focuses on the improvement of the average NMI of partitions. The higher the NMI obtained by the LPAcw is, the better performance the LPAcw has. We should notice that as consensus clusters guide label propagation in the LPAcw, the NMI of the LPAcw relys on the quality of consensus clusters. The higher NMI consensus clusters have, the better stability the LPAcw results have, and vice versa.

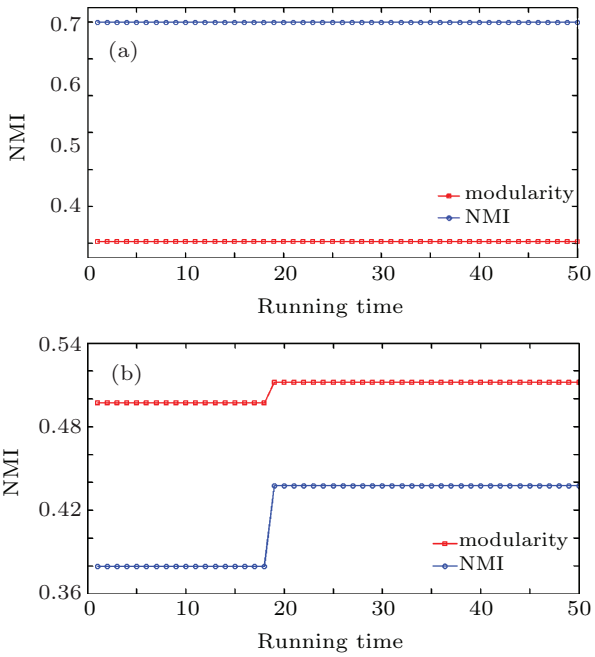


Fig. 5. (color online) The average NMI changes of Karate (a) and Dolphins (b) under different running times. Set proportion parameter $\lambda = 0.6$.

On the other hand, we compare multiple results among LPAcw partitions. Firstly, we use the LPA to obtain 500 consensus clusters of a network, then we select 50 standard clusters from all consensus results, which have a larger modularity among all consensus partitions, and finally we run the LPAcw 50 times to get 50 LPAcw partitions. We compare the NMI among these 50 LPAcw partitions. Figure 5 shows this comparison in Karate and Dolphins. From this comparison, we hope to verify whether the LPAcw can keep good stability if we obtain some good consensus partitions, and this comparison focuses on the stability of the NMI. The more stable the NMI obtained by the LPAcw is, the better stability the LPAcw has, and vice versa.

Modularity We set the consensus iteration as 100, which means applying the basic LPA 100 times to obtain a set of consensus clusters. We set proportion parameter λ and consensus threshold α ranging from 0.1 to 0.7, 0.3 to 0.9 respectively. Figure 6 shows the average modularity obtained by the LPA and LPAcw in Karate and Dolphins. As the consensus cluster, we can see that the LPAcw can get a higher modularity compared with the LPA in the same iteration.

In addition, we also compare the average performance of the LPA and LPAcw on real networks. Table 1 shows maximal modularity and average modularity in real networks obtained by LPA and LPAcw. The value of the LPAcw is selected from

a large number of stable experimental results, and the corresponding value of LPA are collected from Refs. [13], [16], and [27]–[30]. From Table 1, we can find that LPAcw has a better performance than the LPA, whether in maximum modularity or average modularity.

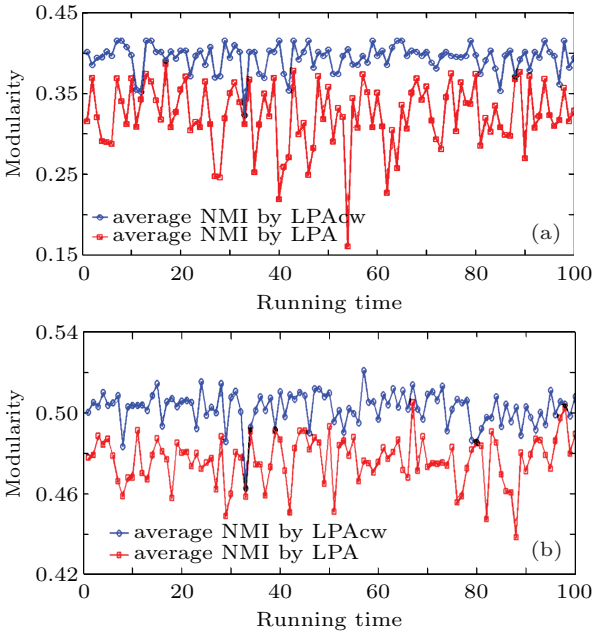


Fig. 6. (color online) The average modularity changes of Karate (a) and Dolphins (b) under different running times.

Table 1. Modularity on real networks.

Network	Q_{max}		$Q_{average}$	
	LPA	LPAcw	LPA	LPAcw
Karate	0.3735	0.4356	0.3502	0.3992
Chesapeake	0.5733	0.5894	0.5610	0.5719
Dolphins	0.516	0.521	0.4864	0.4993
Polbooks	0.5122	0.5286	0.5035	0.5128
Netscience	0.9507	0.9794	0.9125	0.9300
Football	0.5921	0.6190	0.5879	0.5910

Proportion parameter λ As introducing a proportion parameter λ, we hope to know how the consensus weight improves the performance of community partitions. We use the LPA to get 50 consensus clusters and set proportion parameter λ ranging from 0 to 1. For each λ and the same 50 consensus clusters, we run the LPAcw 50 times to show curve changes under different λ. Figure 7 shows the average NMI and average modularity changes under different proportion parameter λ in benchmark networks, and figure 8 plots the average modularity changes under different λ in real networks. When $\lambda = 0$, the LPAcw update label depends on the consensus weight totally, and $\lambda = 1$ the LPAcw equals to the basic label propagation algorithm. From Fig. 8, we can see that in most real networks, the optimal λ is neither 1 nor 0. If we choose a proper proportion parameter λ, it has a better performance than the LPA.

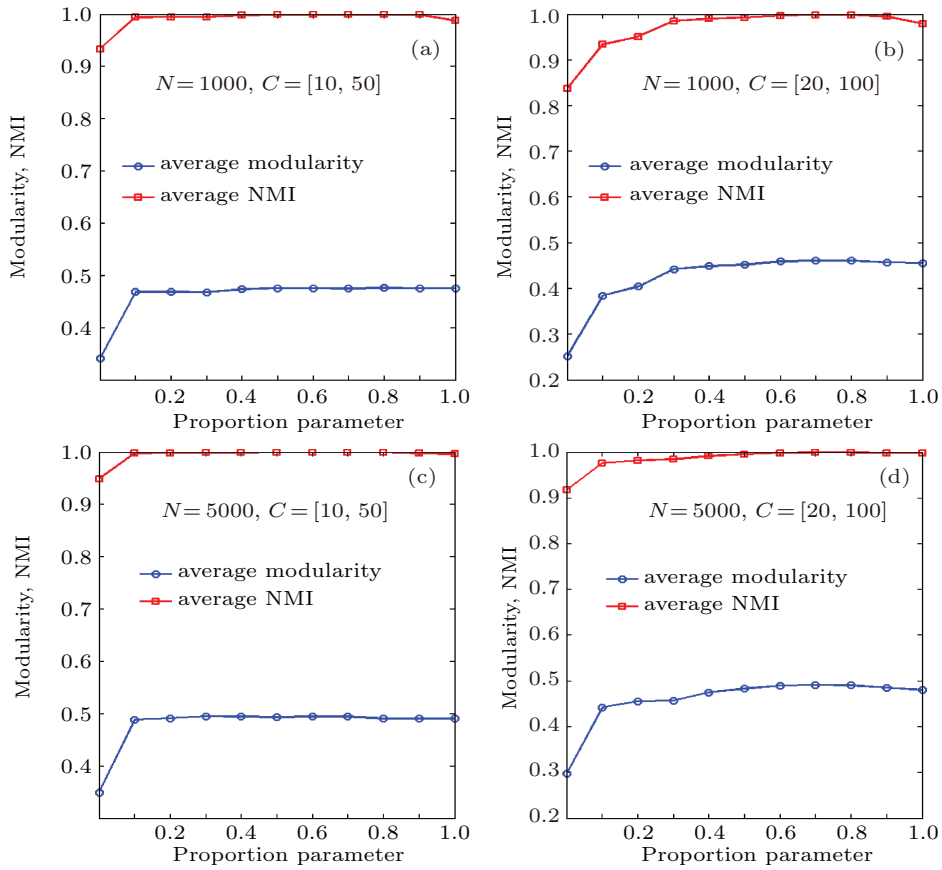


Fig. 7. (color online) The average NMI and average modularity changes obtained by the LPAcw under different proportion parameters on the LFR benchmark. (a) $N = 1000$, $C = [10,50]$; (b) $N = 1000$, $C = [20,100]$; (c) $N = 5000$, $C = [10,50]$; (d) $N = 5000$, $C = [20,100]$.

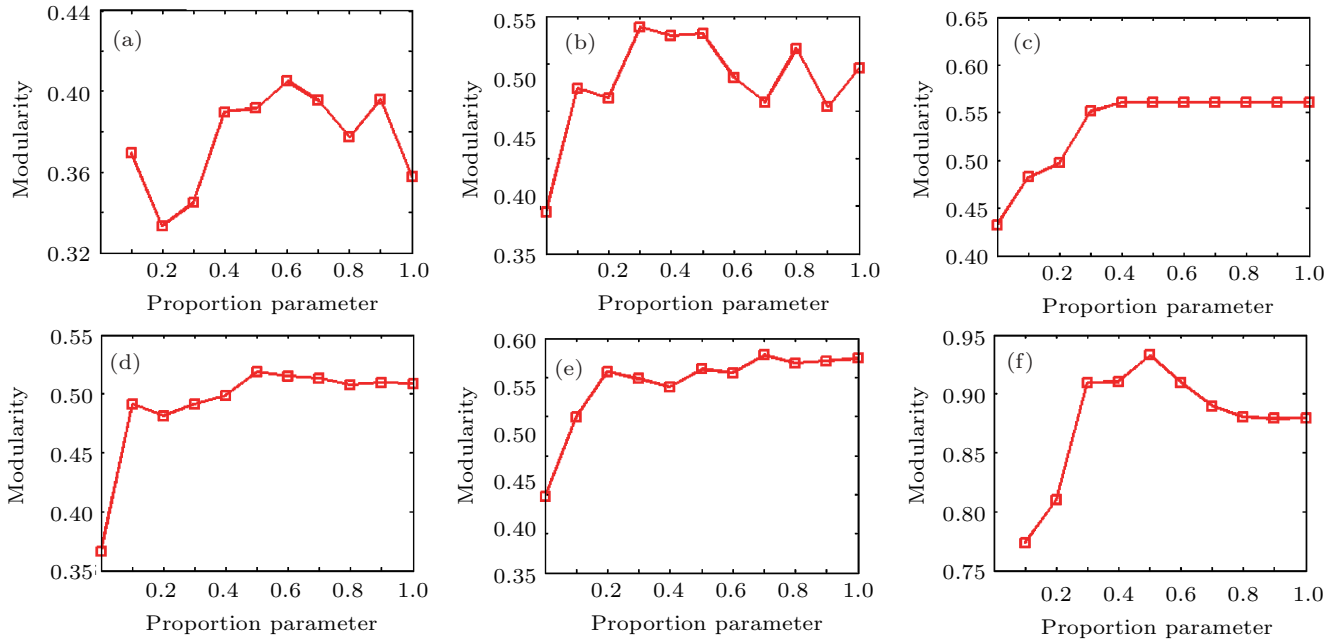


Fig. 8. (color online) The average modularity changes under different proportion parameters in real networks: (a) karate; (b) dolphins; (c) chesapeake; (d) books; (e) football; (f) netscience.

Therefore, we can conclude that the LPAcw is more stable than the LPA and it can also improve the modularity in most real networks, meaning that the LPAcw has a better performance than the LPA.

6. Conclusion

In this paper, we introduce consensus clustering technology into the traditional label propagation algorithm. Combining a proportion parameter λ, we find that in most networks, if we use the consensus weight to guide the label propagation in the community detection process, it has better performance than the basic label propagation algorithm. We believe that this method will be helpful to others working in label propagation research.

References

[1] Girvan M and Newman M E 2002 *Proc. Natl. Acad. Sci. USA* **99** 7821
[2] Seifi M, Junier I, Rouquier J B, Iskrov S and Guillaume J L 2013 *Complex Networks* (Berlin–Heidelberg: Springer) pp. 87–98
[3] Fortunato S 2010 *Phys. Rep.* **486** 75
[4] Fortunato S and Castellano C 2012 *Computational Complexity* (New York: Springer) pp. 490–512
[5] Zou S R, Peng Y J, Liu A F, Xu X L and He D R 2011 *Chin. Phys. B* **20** 018902
[6] Tang S X, Chen L and He Y G 2011 *Chin. Phys. B* **20** 110502
[7] Wang L, Wang J, Shen H W and Cheng X Q 2013 *Chin. Phys. B* **22** 108903
[8] Blondel V D, Guillaume J L, Lambiotte R and Lefebvre E 2008 *J. Stat. Mech.-Theory Exp.* **10** P10008
[9] Newman M E and Girvan M 2004 *Phys. Rev. E* **69** 026113
[10] Brandes U, Delling D, Gaertler M, Gorke R, Hoefer M, Nikoloski Z and Wagner D 2007 *Graph-Theoretic Concepts in Computer Science* (Berlin–Heidelberg: Springer) pp. 121–132
[11] Fortunato S and Barthelemy M 2007 *Proc. Natl. Acad. Sci. USA* **104** 36
[12] Guimera R, Sales-Pardo M and Amaral L A N 2004 *Phys. Rev. E* **70** 025101
[13] Lou H, Li S and Zhao Y 2013 *Physica A* **392** 3095
[14] Raghavan U N, Albert R and Kumara S 2007 *Phys. Rev. E* **76** 036106
[15] Šubelj L and Bajec M 2011 *Phys. Rev. E* **83** 036103
[16] Šubelj L and Bajec M 2011 *Eur. Phys. J. B* **81** 353
[17] Xie J and Szymanski B K 2013 *arXiv.*1303.0868
[18] Topchy A, Jain A K and Punch W 2005 *IEEE Trans. Pattern. Anal. Mach. Intell.* **27** 1866
[19] Goder A and Filkov V 2008 *Alenex* **8** 109
[20] Strehl A and Ghosh J 2003 *J. Mach. Learn. Res.* **3** 583
[21] Lancichinetti A and Fortunato S 2012 *Sci. Rep.* **2** 336
[22] Monti S, Tamayo P, Mesirov J and Golub T 2003 *Mach. Learn.* **52** 91
[23] Radicchi F, Castellano C, Cecconi F, Loreto V and Parisi D 2004 *Proc. Natl. Acad. Sci. USA* **101** 2658
[24] Qinna W and Eric F 2009 ECCS
[25] Danon L, Diaz-Guilera A, Duch J and Arenas A 2005 *J. Stat. Mech.-Theory Exp.* **09** P09008
[26] Liu X and Murata T 2010 *IEEE Second International Conference* 2010 pp. 576–581
[27] Leung I X, Hui P, Lio P and Crowcroft J 2009 *Phys. Rev. E* **79** 066107
[28] Liu X and Murata T 2010 *Physica A* **389** 1493
[29] Cordasco G and Gargano L 2010 *IEEE Int. Workshop Bus. Appl. Soc. Netw. Anal.* pp. 1–8
[30] Shen Y 2013 *Chin. Phys. B* **22** 058903