An improvement of the fast uncovering community algorithm*

Wang Li(王 莉)[a)b)†], Wang Jiang(王 将)[a)b)], Shen Hua-Wei(沈华伟)[b)], and Cheng Xue-Qi(程学旗)[b)]

[a)] College of Computer Science and Technology, Taiyuan University of Technology, Taiyuan 030024, China

[b)] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

(Received 14 May 2013; revised manuscript received 9 June 2013)

Community detection methods have been used in computer, sociology, physics, biology, and brain information science areas. Many methods are based on the optimization of modularity. The algorithm proposed by Blondel *et al*. (Blondel V D, Guillaume J L, Lambiotte R and Lefebvre E 2008 *J. Stat. Mech.* **10** 10008) is one of the most widely used methods because of its good performance, especially in the big data era. In this paper we make some improvements to this algorithm in correctness and performance. By tests we see that different node orders bring different performances and different community structures. We find some node swings in different communities that influence the performance. So we design some strategies on the sweeping order of node to reduce the computing cost made by repetition swing. We introduce a new concept of overlapping degree (OV) that shows the strength of connection between nodes. Three improvement strategies are proposed that are based on constant OV, adaptive OV, and adaptive weighted OV, respectively. Experiments on synthetic datasets and real datasets are made, showing that our improved strategies can improve the performance and correctness.

Keywords: community division, algorithm improvement, performance

PACS: 89.75.Fb, 89.75.Hc **DOI:** 10.1088/1674-1056/22/10/108903

1. Introduction

A common principle is found in nature and human society that individuals always incline to those that are similar. These subgroups were termed communities by Newman *et al*.,[1] where nodes are connected tightly within community and loosely connected among different communities. Finding community is very important for understanding complex network characteristics, discovering latent topics, and predicting network evolution.

The theoretically and practically significant implications have motivated many researchers to study the communities of networks.[2–21] Many effective models are proposed and new applications are exploited. These methods can be roughly classified into six categories: cut size-based, spectral-based, max-flow min-cut-based, centrality- or betweenness-based, physical theory-based, and modularity-based methods.

Cut size-based method is to find the community structure that could optimize the difference between the number or weight-sum of edges inside the communities and that of outside communities.[2] The spectral-based method is a bisection method by calculating the eigenvector and eigenvalue of Laplacian matrix.[3] Max-flow min-cut theorem states the minimum cut between any two vertices and Flake *et al*.[4,5] used maximum flows to identify communities in the World Wide Web. Edge betweenness is the number of shortest paths between all vertex pairs that run along the edge, which shows edge centrality and importance. The betweenness-based method always removes edges with largest betweenness and recalculates betweenness and iteration between these cycles.[6] Some researchers have studied the community structure based on physical theories. A simple model was proposed to generate small-world network with a tunable community structure.[7] Based on the Kuramoto model, they found that increasing the fuzziness of a community structure will markedly enhance the network synchronizability and this network exhibits remarkable synchronization behavior on the topological scale. An adaptive observer[8] was proposed based on the Lyapunov stability theory to identify dynamical network topology. Modularity shows the difference between community topology and stochastic topology with the same degree distribution. Because "Modularity represented one of the first attempts to achieve a first principle understanding of the clustering problem, it is by far the most used and best known quality function",[22] a lot of methods based on modularity were built recently.[9–17] But researchers also found that two challenges exist in this method: one is the fitness problem of topology with large modularity to reality; the other is the performance problem.

For modularity, Fortunato and Barthélemy[23] pointed out that modularity optimization has a resolution limit that may prevent it from detecting clusters which are comparatively small with respect to the graph, even when they are well de-

*Project supported by the Major State Basic Research Development Program of China (Grant Nos. 2013CB329602 and 2012CB316303), the Science Research Foundation for the Returned Overseas Chinese Scholars, China (Grant No. 2010-31), the International Collaborative Project of Shanxi Province, China (Grant No. 2011081034), and the National Natural Science Foundation of China (Grant Nos. 61232010 and 61202215).

†Corresponding author. E-mail: wangli2011@ict.ac.cn

© 2013 Chinese Physical Society and IOP Publishing Ltd

fined communities. A large value for the modularity maximum does not necessarily mean that a graph has community structure.[24,25] Different community topologies may have the same or nearly the same modularity that confuses which one is the real community structure. So, modularity is important for measuring the community topology but it is not enough.

Another challenge is the performance. The scale of networks is becoming enormous due to the rapid growth of information quantity on the Internet. Finding community topology with optimization modularity is non-deterministic polynomial (NP)-complete. Fast and feasible algorithms are urgent. So many heuristic methods are presented.

The first algorithm using maximization modularity as objective function is a greedy agglomerative hierarchical clustering algorithm that was proposed by Newman.[10] Its complexity is $O(n^2)$. Clauset *et al.*[11] pointed out that there are a large number of uselessness operations in Newman's algorithm because of the structure sparsity of the adjacency matrix. They arranged the data in the form of binary tree. Their algorithm complexity is $O(n\log^2 n)$ and it can deal with graph with 10^6 edges. However, this method has a tendency to produce large communities with a large fraction of nodes. Wakita and Tsurumi[15] introduced tricks to balance the sizes of communities in order to avoid this undesired effect. Xiang *et al.*[16] proposed a new metric to quantify the structural similarity between subgraphs and designed an algorithm for community identification based on the metric. A novel greedy approach has been proposed by Blondel *et al.*[17] The algorithm can fast unfold communities in large networks with some high performances. It could deal with graphs with 10^9 edges in a reasonable time and the result has high modularity. So this algorithm was widely used recently. However, with its wide application, many researchers find that its performance changes with the order of sequential sweep over vertices. What makes the sweeping order influence its performance? What can be done to improve the performance? These are the two goals of this paper.

2. Improved fast unfolding community algorithm

2.1. Blondel's fast unfolding community method

The fast method of unfolding communities, proposed by Blondel *et al.*,[17] is for a weighted graph. It consists of two iterative phases.

The first phase is community merging. Initially assign a different community to a different node. For each node i, evaluate the gain of modularity if it is moved from its community to the community of its neighbors. Choose the neighbor node j, if it makes the modularity gain a maximum, and it is also positive. Then move i to the community of j, otherwise i stays in its original community. Repeat this process for all nodes till no further improvement can be achieved.

The second phase is graph reconstruction. Compact the communities into nodes and assign weights to the links between the new nodes. Their weights are the sum of weights of the links between nodes in the corresponding two communities. Then a new weighted network is constructed.

Reapply the first phase to this new network and iterate it till no high modularity can be obtained.

We take the first phase and second phase as one "pass" and repeat several "passes" till the modularity reaches a maximum.

This algorithm has high performance and it is easy to implement on large networks. It is one of the most popular methods of unfolding communities.

2.2. Algorithm analysis

Just as pointed out in Ref. [17], we also find that the order of sequential sweep over nodes influences the running time. We do some experiments on benchmark network and real network with middle size. Figure 1 shows the relationship between sweep order and performance.

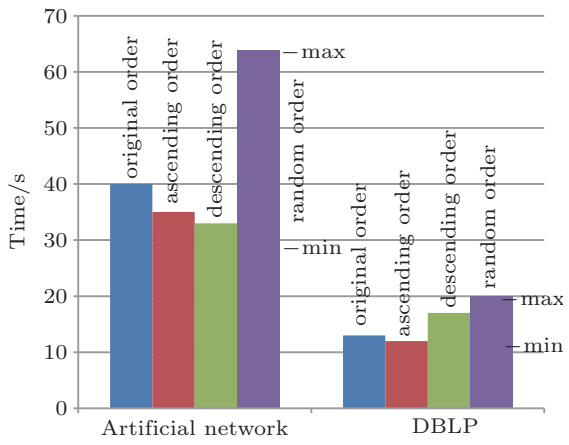


Fig. 1. (color online) Different running times by different sweep orders.

We can see that different orders indeed bring different performances. Sometimes the influence is big, sometimes it is small.

Analyzing the algorithm process, we see that the first phase takes the most running time and after the first phase the community number dramatically descends and modularity dramatically increases. In the first process, some nodes frequently change their community membership role and this expends much unnecessary running time. We find that these nodes always do not have close relationship with their neighbor nodes and different local relationships make them moved

to different local communities. Figure 2 shows an example of this node.

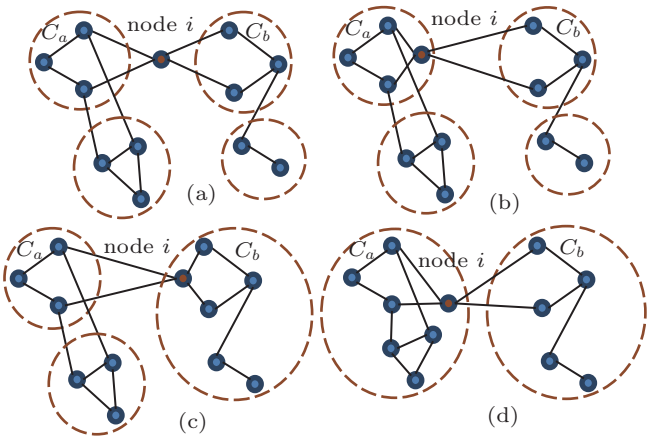


Fig. 2. (color online) Demonstration of swing nodes.

In Fig. 2, Node i which is termed swing node has two neighbor communities C_a and C_b. The relationships between C_a and swing node i, C_b and swing node i are nearly the same. In Blondel's algorithm, node i may move to the community C_a in the first iteration, in the second iteration it may move to community C_b, and later may move back to community C_a. Its swing takes unnecessary running time.

We give definitions to formally represent their characteristics.

Definition 1: Overlapping degree (OV) OV denotes the overlapping degree of two nodes' neighbors, which shows the connection strength between nodes. There are two different measurement methods of overlapping degree. One is based on the weight sum of edges between nodes and shared neighborhoods. The other one is based on the size of shared neighborhood. So we give two definitions to calculate the nodes connection strength. One is $\mathrm{OV}_{\mathrm{weight}}$ that is based on the weight. For nodes i and j, their $\mathrm{OV}_{\mathrm{weight}}$ values can be calculated by the following formula

$$\mathrm{OV}_{\mathrm{weight}}(i,j)=\frac{\sum_{p=1,w_p\in N_i\cap N_j}^{k}(\mathrm{weight}(i,w_p)+\mathrm{weight}(j,w_p))}{\sum_{p=1,w_p\in N_i\cup N_j}^{K}(\mathrm{weight}(i,w_p)+\mathrm{weight}(j,w_p))}. \tag{1}$$

The other alternative one is $\mathrm{OV}_{\mathrm{node}}$ that shows the ratio of shared neighborhood size to the size sum of both neighborhoods. We define it as follows:

$$\mathrm{OV}_{\mathrm{node}}(i,j)=\frac{|N_i\cap N_j|}{|N_i\cup N_j|}, \tag{2}$$

where N_i is the neighbor set of node i. The weight (i,j) is the weight value of edge (i,j). $|R|$ is the node number of set R.

For convenience, we use OV to represent both $\mathrm{OV}_{\mathrm{weight}}$ and $\mathrm{OV}_{\mathrm{node}}$ in this paper.

Definition 2: Swing nodes Swing nodes are those nodes that have no clear community structure with their neighbors. That means the connection strength between them and their neighbors are all smaller than some threshold value,

$$\mathrm{OV}(i,j)<\theta,\quad j\in\mathrm{NeighSet}(i), \tag{3}$$

where NeighSet(i) is the neighborhood set of node i, and θ is a threshold.

Formula (3) is the characteristic representation of quantity for swing node i.

We have the intuition that if we deal with swing nodes later the unnecessary moving and computing will be reduced and the running time maybe decreases. However, discriminating which one is swing node is costly because we have to compute nodes and all their neighbors. Now we consider modularity Q (formula (4)). For node i, the degree of its neighbor node j is smaller, and their modularity is bigger. Algorithm of Blondel's aims at increasing modularity to maximum, then node will have a bigger probability to move to the community of its neighbor with a least weight-sum. So when we determine which one is swing node we do not calculate all neighbors, we just calculate the OV of node i and its neighbor with the least weight-sum (if in unweighted graph, it is the neighbor with least degree)

$$Q=\frac{1}{2m}\sum_{i,j}\left[A_{ij}-\frac{k_ik_j}{2m}\right]\delta(c_i,c_j). \tag{4}$$

Therefore we simplify the definition of swing node as follows.

Definition 2′: Swing nodes Node i is a swing node, iff

$$\begin{aligned}&\mathrm{OV}(i,j)<\theta,\ j=\mathrm{argmin}\ \mathrm{weighted_sum}(k)_{k\in\mathrm{NeighSet}(i)}\\&\text{if}\ \ \mathrm{OV}(i,j)\ \text{is}\ \mathrm{OV}_{\mathrm{weigt}}(i,j)\\&j=\mathrm{argmin}\ \mathrm{degree}(k)_{k\in\mathrm{NeighSet}(i)}\\&\text{if}\ \mathrm{OV}(i,j)\ \text{is}\ \mathrm{OV}_{\mathrm{node}}(i,j)\end{aligned} \tag{5}$$

$0<\theta\leqslant 1$. θ is a thresholod value.

It is obvious that the value of θ is important, which determines the effect of the improved algorithm.

In Blondel's algorithm, we find the whole OV rises with "pass". So the initial threshold θ may take effect for reducing unnecessary computing in the former "pass", but in the later the OVs of all pairs increase, initial threshold may become useless. So we design an automatic strategy (AS) to assign the threshold value.

(i) AS-strategy and WAS-strategy

In this strategy, we give one initial value. With the procedure it will adaptively change. The update method is

$$\theta_0 = \text{constant}, \quad \theta_i = \frac{\theta_{i-1} + \mathrm{OV}(i,j)}{2}. \tag{6}$$

Because there are two different methods of OV, we have two different automatic strategies. One is based on $\mathrm{OV}_{\mathrm{weight}}$ and the other is based on $\mathrm{OV}_{\mathrm{node}}$. We distinguish the two strategies and refer to $\mathrm{OV}_{\mathrm{node}}$-based strategy as AS-strategy, $\mathrm{OV}_{\mathrm{weight}}$-based strategy as WAS-strategy.

OV can help reduce unnecessary computing, but meanwhile it takes extra computing task into the process. Besides computing OV, AS takes more additional computing task on adaptive threshold and there is a tradeoff between OV, AS, and total performance. So we design another strategy for threshold θ. This strategy also delays the swing nodes but it simplifies the computing of OV threshold value. It is a constant threshold-based strategy.

(ii) Cons-strategy

This strategy is to assign constant initial and increment value in advance based on experience (We refer to this strategy as Cons). Compared with AS, this strategy reduces the process of modifying θ at each node. From several tests on benchmark and real datasets, we find 0.2 is a suitable initial value. And with the pass, we set it to be an increment constant value 0.2. Because we do not know how many iterations will happen, θ may be bigger than 1. We set it to be 1 when θ is equal to or bigger than 1.

```
Input: Network(Node,Edge)
       or Network(Node,Edge,Weight)
output: C1,C2,C3, ···, Cn                    //  communities topology
Begin
  constant→θ
  Do
    {//FirstPhase()
      For each i∈Node
        Calculate  OV_ij  j=argmin degree(k)_{k∈NeighSet(i)}
                   or  j=argmin weighted_sum(k)_{k∈NeighSet(i)}
        If (OV_ij > θ )
        n_j → prior_order;
        Else
        n_j → delay_order;
        Endif
      Endfor
      prior_order ∪ delay_order→OV_order
      Do
       For each i∈OV_order
        i∪C_j→C_j,   j∈NeighSet(i)&&Max (ΔModularity)_{i→C_j}
       Endfor
      until the Modularity is Maximum;
    //SecondPhase
    Compact each community to one new node and build new network;
    update θ;
    }
  Until the whole Modularity is Maximum
End
```

Fig. 3. Pseudocode of improved algorithm.

2.3. Improved algorithm

Comparing with the original algorithm, we add preprocedure for reordering nodes. If the $\mathrm{OV}(i,j)$ is smaller than threshold θ, move it to the delay node set, otherwise do it. The pseudocode of improved algorithm is given as follows.

3. Experiments and analysis

To verify our improved algorithms, we do experiments on artificial benchmark datasets and real networks. We choose different size networks to evaluate our improved algorithms. There are four algorithms evaluated. They are the original algorithm with the original order, improved algorithm with AS strategy, improved algorithm with Cons strategy, and improved algorithm with WAS strategy. And their abbreviations respectively are Blo, Blo–AS, Blo–Cons, and Blo–WAS. The evaluation standard consists of three aspects: correctness, modularity, and performance.

Correctness shows the fitness between unfolded community structure and real community structure. The normalized mutual information (NMI)[27] is widely used to measure the similarity between two partitions based on information theory, which has been proved to be reliable.

Our experiments are done in the same hardware and running environment: double CPU, Linux Ubuntu 32bit, with a memory of 3G.

3.1. Tests on artificial networks

LFR benchmark[28,29] is a widely used network generator for network analysis. We use LFR to build different size benchmark networks. For artificial networks, we can obtain the real community structure, so we can evaluate not only the modularity and performance, but also the correctness of the result.

3.1.1. Correctness, performance, and modularity evaluation with different network scales

We generate different scale benchmark networks from 10 thousand to 2 million. The parameters are $k = 15$, $\max k = 50$, and $\mu = 0.6$. The generated networks are well community structures based on these parameters. For one benchmark network, we do the same algorithm 10 times and average their modularity values, NMIs and running times as the final results. The results are compared in Fig. 4.

We can see that the NMIs and modularity values of Blon–AS, Blon–WAS, and Blon–Cons are all some higher than those of original algorithm. But the difference is some small. These mean that the NMI is mainly determined by the original algorithm and our improvements on the sweeping order do not greatly influence the result quality. With the increase of network size, the running time increases. In whatever size networks, Blon–AS and Blon–Cons both behave better than others. Because Blon–WAS needs more time to calculate the OV and adjust threshold for each node, it does not behave very well.

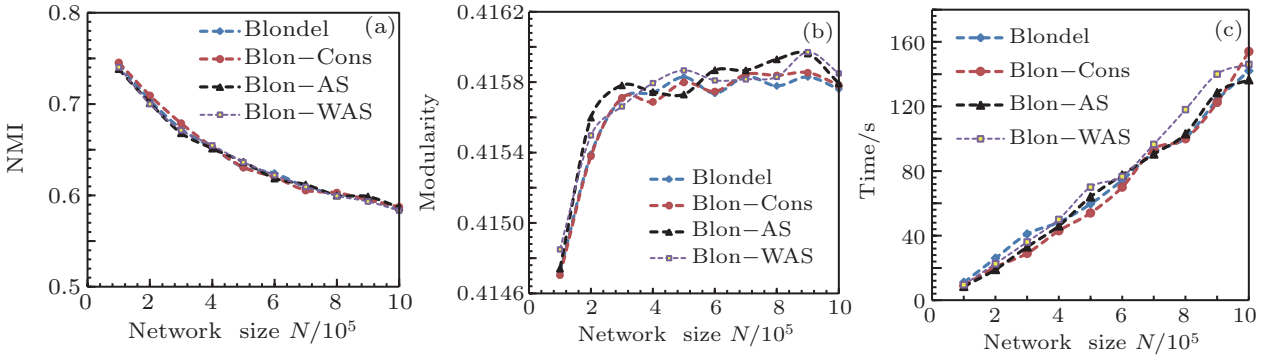


Fig. 4. (color online) (a) Comparison among the correctnesses for different algorithms on different network scales. (b) Comparisons among the modularity values for different algorithms on different network scales. (c) Comparisons among running times for different algorithms with different network scales.

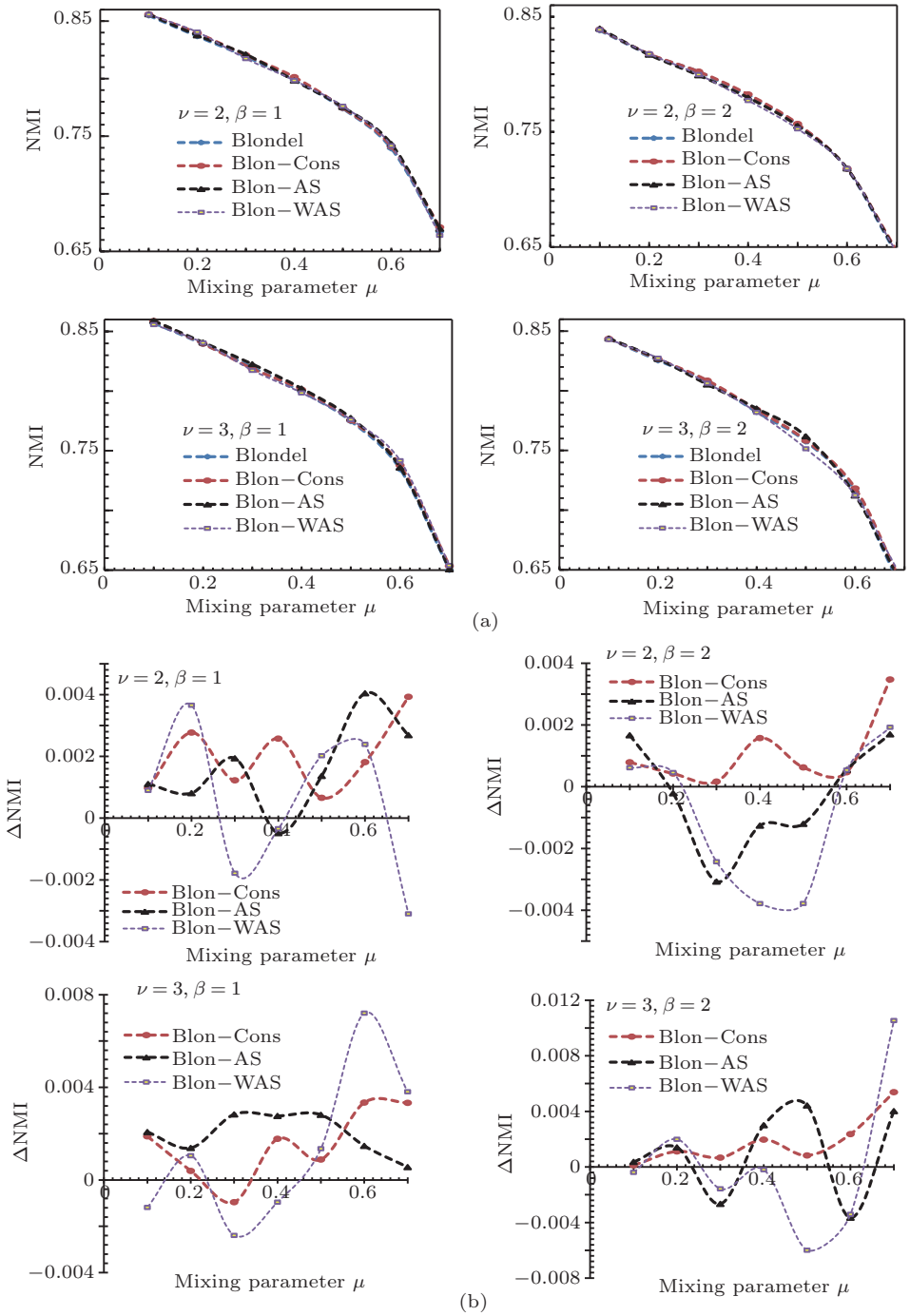


Fig. 5. (color online) NMI comparisons with and NMI differences from those of original algorithm, Panel (a) shows NMI in different networks with different parameters. Panel (b) displays NMI differences between the improved algorithms and the original algorithm.

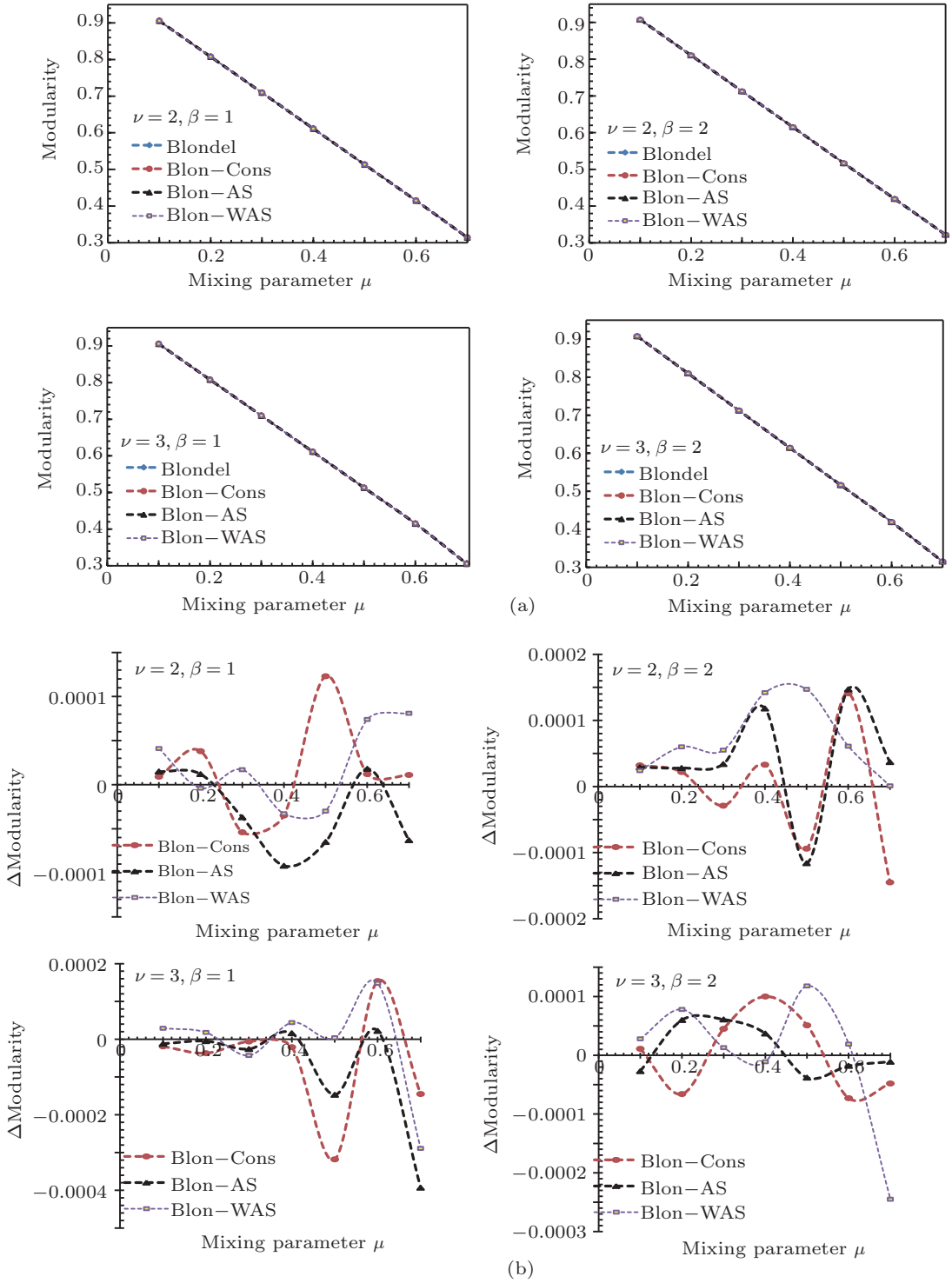


Fig. 6. (color online) Comparisons in modularity among different algorithms with different values of μ. Panel (a) shows modularity values in different networks with different parameters. Panel (b) displays the modularity differences between the improved algorithms and the original algorithm.

3.1.2. The quality evaluation of unfolding communities with different mixing parameters

We also focus on the quality evaluation of unfolding communities with different mixing parameters. In the real world, network systems are complex and not every network has a latent community structure or a clear community structure. A good community detection method should find not only the community structure with maximum modularity, but also truly community structure. That is, if there are not latent community structures in network, the unfolding algorithms should result in a low modularity.

In LFR, mixing parameter μ controls the latent community structure. The bigger the value of μ, the fuzzier the community structure is. The smaller the value of μ, the better the

community structure is. We can generate benchmark networks to simulate the complex networks. We set the value of μ to be in a range from 0.1 to 0.7 in intervals of 0.1. The node number is 100 thousand. We make the comparisons of modularity and NMI among different values of μ. We do each algorithm 10 times and average their modularity values and NMIs. The comparisons are made in Figs. 5 and 6.

As mixing parameter increases, the NMIs of all algorithms decrease, especially when mixing parameter is bigger than 0.6, NMI suddenly drops. To clearly observe the NMI difference, we use the NMI of original algorithm as baseline and calculate the NMI difference between improved algorithm and original and the results show in Fig. 5(b). Blon–Cons actives stably and it has higher NMI than original algorithm. Blon–AS and Blon–WAS are active unstably in NMI quality.

The four algorithms almost have the same modularity. We also make the modularity difference based on the modularity of original algorithm as shown in Fig. 6(b).

The Δ modularity values of improved algorithms are not monotonic. When mixing parameter is bigger than 0.6, all Δ modularity values are all smaller than 0, which shows that the improved algorithms help unfold the latent topology, especially when the network has no community structure.

3.2. Tests on real datasets

In order to confirm the validity of our improved algorithms, we select three real networks with different scales from 30 thousand to 300 thousand (http://snap.Stanford.edu/data). These datasets are free datasets and they also have true community structures. So we can verify our result correctness, modularity, and performance. These datasets are from three different social networks: DBLP, Youtube, and LiveJournal. The three networks are all undirected unweighted networks, which are special case of weighted graph.

The first dataset is extracted from the computer science in DBLP that shows the coauthor relationship. The second dataset is extracted from Youtube. Youtube is a video-sharing social network, in which nodes are users and edges are friendships. The third dataset is extracted from LiveJournal. It is a free blogging community where users describe friendship to each other.

The details of the three datasets are shown in Table 1.

Table 1. Detailed information about datasets.

	Nodes	Edges	Type
DBLP	317080	1049866	Collaborative network
Youtube	1134890	2987624	Social network
LiveJournal	3997962	34681189	Social network

We implemented the four algorithms on these networks, and the measurements of correctness, modularity, and performance of the different algorithms are given in the following Tables 2, 3, and 4.

Table 2. Results on DBLP.

	Time/s	Modularity	NMI
Blondel	13	0.81923	0.083433
Blon–Cons	10	0.820455	0.084242
Blon–AS	12	0.820927	0.080859
Blon–WAS	12	0.820444	0.081146

Table 3. Results on Youtube.

	Time/s	Modularity	NMI
Blondel	39	0.710458	0.075517
Blon–Cons	34	0.712148	0.075829
Blon–AS	36	0.716044	0.079491
Blon–WAS	55	0.713974	0.078876

Table 4. Results on LiveJournal.

	Time/s	Modularity
Blondel	155	0.714564
Blon–Cons	154	0.716239
Blon–AS	309	0.744835
Blon–WAS	453	0.745997

The NMI of the result from Blon–Cons is higher than those from others on the DBLP dataset. On the Youtube dataset, the NMI of result from Blon-As is higher than others. The outcome is the same as that of benchmark networks. This shows that our improvement algorithms can improve the correctness of this algorithm to some extent.

We can see that the modularity values of improved algorithms in 3 datasets are higher than that of original algorithm, which shows that adjusting the sweeping order helps to maximize the modularity. However, we also see that the advantage is not very big.

For the performance aspect, we can see that Blon–Cons behaves best and it runs almost 15%$\sim$30% higher than original algorithm. Blon–AS and Blon–WAS have to cost more running time to calculate OV and threshold than Blon–Cons. So when the network scale is bigger, their performances become bad.

4. Conclusion

In this paper we make some improvements on the fast uncovering community algorithm designed by Blondel *et al*. We design some strategies on the sweeping order of node to reduce the computing cost made by repetition swing. We introduce a new concept OV that shows the strength of connection between nodes. Three improvement strategies are proposed that are based on constant OV, adaptive OV, and adaptive weighted OV. Experiments on synthetic datasets and real datasets are made, which shows our improved strategy can improve the performance.

References

[1] Newman M E J and Girvan M 2004 *Phys. Rev. E* **69** 026113
[2] Kernighan B W and Lin S 1970 *Bell System Teeh.* **49** 291
[3] Barnes E R 1982 *SIAM J. Algebr. Discrete Methods* **3** 541
[4] Flake G W, Lawrence S and Giles C L 2000 *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 20–23, 2000, Boston, USA, p. 150
[5] Flake G W, Lawrence S, Giles C L and Coetzee F M 2002 *IEEE Computer* **35** 66
[6] Girvan M and Newman M.E.J 2002 *Proc. Natl. Acad. Sci. USA* **99** 7821
[7] Wang X H, Jiao L C and Wu J S 2010 *Chin. Phys. B* **19** 020501
[8] Fan C X, Wan Y H and Jiang G P 2012 *Chin. Phys. B* **21** 020510
[9] Cheng X Q and Shen H W 2011 *Complex Systems and Complexity Science* **8** 46 (in Chinese)
[10] Newman M E J 2004 *Phys. Rev. E* **69** 066133
[11] Clauset A, Newman M E J and Moore C 2004 *Phys. Rev. E* **70** 066111
[12] Danon L, Díaz-Guilera A and Arenas A 2006 *J. Stat. Mech.* **11** 11010
[13] Pujol J M, Béjar J and Delgado J 2006 *Phys. Rev. E* **74** 016107
[14] Du H, Feldman M W, Li S and Jin X 2007 *Complexity* **12** 53
[15] Wakita K and Tsurumi T 2008 *Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science* **5211** 408
[16] Xiang B, Chen E H and Zhou T 2009 *Studies in Computational Intelligence* **207** 73
[17] Blondel V D, Guillaume J L, Lambiotte R and Lefebvre E 2008 *J. Stat. Mech.* **10** 10008
[18] Chen D, Shang M, Lü Z and Fu Y 2010 *Physica A* **389** 4177
[19] Li W 2011 *Expert Systems with Applications* **38** 94
[20] Shen H W, Cheng X Q, Cai K and Hu M B 2009 *Physica A* **388** 1706
[21] Li W, Bi Y J, Wu W L, Lian B and Xu W 2013 *International Computing and Combinatorics Conference* (COCOON), June 21–23, Hangzhou, China, p. 821
[22] Santo F and Claudio C 2009 *Encyclopedia of Complexity and Systems Science* **25** 1141
[23] Fortunato S and Barthélemy M 2007 *Proc. Natl. Acad. Sci. USA* **104** 36
[24] Guimer'a R, Pardo S M and Amaral L A N 2004 *Phys. Rev. E* **70** 025101
[25] Andrea L and Santo F 2011 *Phys. Rev. E* **84** 066122
[26] Newman M E J 2006 *Proc. Natl. Acad. Sci. USA* **103** 8577
[27] Danon L, Duch L, Guilera A D and Arenas A 2005 *J. Stat. Mech.* **9** P09008
[28] Lancichinetti A, Fortunato S and Radicchi F 2008 *Phys. Rev. E* **78** 046110
[29] Lancichinetti A and Fortunato S 2009 *Phys. Rev. E* **80** 016118