A Hierarchical Diffusion Algorithm for Community Detection in Social Networks

Keyi Shen, Li Song, Xiaokang Yang, Wenjun Zhang
Institute of Image Communication and Information Processing,
Shanghai Key Laboratory of Digital Media Processing and Transmissions,
Shanghai Jiao Tong University
shenkeyi@sjtu.edu.cn, song_li@sjtu.edu.cn, xkyang@sjtu.edu.cn, zhangwenjun@sjtu.edu.cn

***Abstract*—Community discovery is one of the most important steps to understand the social networks. We propose a hierarchical diffusion method to detect the community structure. Our algorithm is based on the idea that people in different communities usually share less common friends. We also make use of the fact that people usually make decisions based others' choices, especially their friends'. Our algorithm can distinguish between pseudo-communities and meaningful ones. Tests on both classical and synthetic benchmarks show that our algorithm is comparable to state-of-the-art community detection algorithms in both computational complexity and accuracy measured by the so-called normalized mutual information.**

***Keywords*—social networks; threshold; diffusion;**

I. Introduction

Social networks have aroused enormous amount of interest of different disciplines. Social network data collected from different sources, such as online social network sites, blogs, emails and mobile phone networks etc., can be simply modeled as graph with nodes and edges, in which nodes represent agents and nodes are connected if there is some social interaction between them.

Although there is no strict definition of community, it is important to extract the community structure from social networks. Qualitatively, a community is defined as a set of nodes which are densely connected, therefore nodes in different communities have sparser interconnections. Community might be synonymous with cluster, group, module under different contexts. Community structure is an interesting feature of social networks. Graphs generated randomly usually do not have this feature. The goal of the community detection algorithm is to partition the network into communities, which means to assign each node a community ID. So it's closely related to the graph partition and clustering problem.

Online social networks usually contain enormous numbers of nodes and edges. Our method is based on a local measure of similarity and the diffusion of membership of different communities. The simplicity of our algorithm makes it possible to extract the community structure from a huge social network with both accuracy and efficiency.

Whether there really exists community structure in a given social network is a question overlooked by many algorithms. They have a tendency to partition networks into modules in any cases. Our algorithm will not get a partition if no community structure is found. It will stop, when the network contains no smaller scale of community structure, instead of iterating until each node becomes one community. This feature enables us to adopt a hierarchical framework and to disclose the community structures at different scales.

II. Related Work

Before we introduce different kinds of community detection algorithms, it is necessary to give some criteria to quantify the quality of their partitions. If the real community structure of the network is known, we can use the normalized mutual information (NMI) as a measure to compare the communities found by the algorithm with the natural communities[1], [2], [3]. The normalized mutual information is derived from the information theory, and is defined as:

$$NMI(P_1, P_2) = \frac{2(H(P_1) + H(P_2) - H(P_1, P_2))}{H(P_1) + H(P_2)}. \quad (1)$$

Where $H(P_i)$ is the entropy of the partition P_i, $i = 1, 2$, $H(P_1, P_2)$ is the joint entropy. The NMI ranges from 0 to 1. If the partition we get P_1 is identical to the real partition P_2, the NMI equals to 1. If P_1 and P_2 are independent of each other, the NMI equals to zero, which means that having complete knowledge of P_1, we still know nothing about the real partition P_2[1].

Unfortunately, few networks have convincing community structure information. So modularity was introduced. The definition of modularity Q of an unweighted and undirected network is given in (2)[4], [5].

$$Q = \sum_i (e_{ii} - b_i^2) \quad (2)$$

Where e_{ij} is the fraction of endpoints of edges in community i for which the other endpoint of edge lies in community j, and $b_i = \sum_j e_{ij}$ is the fraction of all endpoints of edges that lie in community i[5]. Modularity compares the fraction of the within-community edges with that of a randomly connected graph. Q varies between 0 and 1, where $Q = 0$ corresponds to random networks without community structure, and Q with relatively large value implies the existence of strong community structure.

There are two basic approaches to detect the community structure: agglomerative methods and divisive methods[6]. In an agglomerative algorithm, the weight of every edge is

978-0-7695-4235-5/10 $26.00 © 2010 IEEE
DOI 10.1109/CyberC.2010.57

calculated at first. The weight can be regarded as a measure of similarity between different nodes. We start from a graph with all its nodes and no edges, and edges ordered by their weight are added iteratively. In this way nodes can be agglomerated to larger and larger communities. In the framework of a division algorithm, we start from the whole graph and iteratively remove the edges with least similarity, therefore divide the network into several components.

Since modularity can measure the quality of a partition without the knowledge of the real community structure, many algorithms adopt it as the objective function and detect communities by maximizing it. The optimal modularity is hard to get, since it cannot be calculated until a partition is given. Actually it is proven to be NP-complete[7]. Many modularity-based algorithms, such as greedy methods[8], spectral methods[9], simulated annealing[10], or extremal optimization[11], turn to seek the sub-optimal modularity. The main drawback of these modularity-based methods is the resolution limit problem[12]. They might fail to find some communities even when the community structure is well-defined.

Louvain method[13] is a hierarchical agglomerative algorithm based on the optimization of modularity. At first, every node is a community itself. Then every node will choose to quit from current community and join one of its neighbor's community, if it brings some modularity gains. This step will be iterated until no nodes want to change their community ID. Next, nodes in the same community will merge into one node, and yield a smaller and weighted network. Above procedure is iterated until no more modularity gains are available. Louvain method outperforms other methods in terms of the running time.

Arrays of algorithms tackle the same problem from different point of views. The Potts model algorithm is based on the optimization of the Hamiltonian of a Potts-like spin model, where the spin state represents the community ID of the node[14]. The method Infomap introduced in [15] converts the partition problem into a problem of optimally compressing the information on the structure of graph. It divides the network by compressing a description of the probability flow. More specifically, it takes the minimum description length of random walk, which is the dynamic process of the graph, and optimizes this objective function by a combination of greedy search and simulated annealing.

The clustering coefficient of a node u is defined as the probability that two randomly selected friends of u are friends with each other. Ravasz et al. have proposed a method based on the clustering coefficient[16]. In our method, we only adopt the numerator of the clustering coefficient as a similarity measure, which leads to quite different outcomes. Radicchi et al. have proposed an agglomerative method based on a similar clustering coefficient known as the edge-clustering coefficient[17]. There is also a duality between the similarity measure we adopt and the node's clustering coefficient used in [18].

More complete reviews and the comparative analysis of different algorithms can be found in [5], [19], [20].

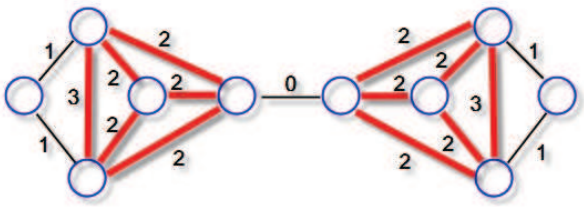


Figure 1. Illustration of the thresholding step of our algorithm

III. Algorithm Description

An undirected and unweighted social graph $G(V, E)$ with N nodes is considered in our algorithm. After processed by the following procedure, a partition P of G is returned, where $P = \{G_1(V_1, E_1), ..., G_m(V_m, E_m)\}$, $V_1 \cup V_2 \cup ... \cup V_m = V$, and every node $u \in V$ will get a community ID $C(u)$. $C(u) = i$, if node $u \in V_i$. The initial value of the community ID is zero for every node. The proposed hierarchical diffusion algorithm includes two major steps.

The first step is the thresholding. We define $N(u, v)$ as the number of common neighbors of node u and v. We use $N(u, v)$ as a local measure of similarity between node u and v, and denote the weight of $edge(u, v)$ as $W(u, v)$, and let $W(u, v) = N(u, v) + 1$, if u and v is connected. The weight of each edge is calculated at first.

We then set a threshold Th and get the $G(Th)$, where $G(Th) = G(V(Th), E(Th))$, $V(Th) = \{u : \exists v \in V, W(u, v) \geq Th\}$, $E(Th) = \{edge(u, v) : W(u, v) \geq Th\}$. We call the nodes in $V(Th)$ the *core members* of their communities. Suppose that $G(Th)$ has m connected components, the community ID $C(u)$ of the core member u in the ith component will be i, $i = 1, 2, ..., m$.

An illustration of the thresholding step is shown in Fig. 1, where each $edge(u, v)$ is labeled by the corresponding $N(u, v)$. The value of Th is 3. The edges in $E(3)$ are highlighted in red and bold. The core members are the endpoints of these red bold edges. In this example, $G(3)$ has 2 components. We can assign the core members in component of the left side to community 1, and assign those in the other component to community 2.

The second step is diffusion. We define $Z(u)$, the total weight of node u, in (3).

$$Z(u) = \sum_{k=1}^{d(u)} W(u, nb_k(u)) \tag{3}$$

Where $nb_k(u)$ is the kth neighbor of node u, $d(u)$ is the degree of node u.

Then we define $Z_m(u)$, the total weight between node u and nodes in community m in (4).

$$Z_m(u) = \sum_{k=1}^{d(u)} W(u, nb_k(u))\delta(C(nb_k(u)), m) \tag{4}$$

Where $\delta(i,j)$ is the Kronecker delta function, i.e. $\delta(i,j)=1$, if $i=j$, and $\delta(i,j)=0$, if $i\neq j$.

Starting from the core members' neighbors, membership of community spreads to the nodes satisfying (5).

$$Z_m(u) > \frac{1}{2}Z(u) \tag{5}$$

If (5) holds, node u will be allocated to community m, i.e. $C(u)=m(m>0)$. We call all nodes with $C(u)>0$, at the current stage, the *strong members* of their communities.

Diffusion will not stop until no new strong member can be found. Then starting from the strong members' neighbors, we assign $C(u)$ according to (6).

$$C(u) = \arg_m \max_{m>0} Z_m(u) \tag{6}$$

This second round of diffusion makes sure that every node has joined a community. In a special case, node u has equal chances to join community m_1 and m_2, i.e. $Z_{m_1}(u) = Z_{m_2}(u) > Z_{m_i}(u)$, where $m_i \neq m_1$, $m_i \neq m_2$. We assign this node randomly. It is also possible to allow this node to join both two communities, which makes our algorithm capable of detecting some simple overlapping communities.

Fig. 2 shows the pseudocode of our algorithm.

If the thresholding step only get one component from $G(Th)$, there is no need to execute the diffusion step, therefore no division is made. If the number of sub-graphs in partition P is one, that means no community structure is found, our algorithm will stop. Otherwise, all sub-graphs V_i will be taken as input networks and processed by our algorithm iteratively. This iteration will continue until no partition is available, which makes our algorithm a hierarchical one.

Algorithm:
Input: Graph $G=(V,E)$
Initialization;
//The Thresholding Step;
for every edge $(u,v)\in E$ **do**
  for every node $\in Neighbor(u)\cap Neighbor(v)$ **do**
    Update $N(u,v)$ and $W(u,v)$;
  end for
end for
set Threshold Th;
for every edge $(u,v)\in E$ **do**
  if $W(u,v)\geq Th$ **then**
    Push node u and v into $V(Th)$;
    Push $edge(u,v)$ into $E(Th)$;
  end if
end for
for every edge $(u,v)\in E$ **do**
  Get $C(u)$ according to $G(Th)$;
end for
//The Diffusion Step;
for every node u **do**
  for every node $v\in Neighbor(u)$ **do**
    $m=C(v)$;
    Update $Z_m(u)$;
    if $Z_m(u)>\frac{1}{2}Z(u)$ **then**
      $C(u)=m$;
      break;
    end if
  end for
end for
while $\exists$ node u with $C(u)=0$ **do**
  Get $C(u)$ according to (6);
end while
Get P according to $C(u)$;
Output: $C(u)$, Partition P

Figure 2. Our algorithm for community detection

IV. Algorithm discussion

In this section the relationship between our algorithm and some classic sociological findings is presented. The computational complexity of our algorithm, which is related to the choice of threshold Th, is also discussed.

A. Sociological Explanation

The similarity measure $N(u,v)$ of node u and v is supported by the triadic closure principle, which suggests that in a social network, there is an increased likelihood that two people will become friends if they have friends in common. So it's reasonable to count the number of triangles in a social network. The number of common friends of node u and v is related with both the neighbors of node u and the neighbors of u's neighbors. Our method makes use of these secondary information instead of only using the basic degree information, which makes it possible to understand the social networks better.

We don't normalize the weight between two nodes. If we normalized our weight by the number of total neighbors, some small weight with a smaller denominator would become the core members first after a division operation. In our method, those nodes with large weight are considered as more influential nodes, and nodes with relatively small weight will only have a chance to become the core members of sub-communities according to our hierarchical scheme.

In social network analysis, an $edge(u,v)$ in graph G is a bridge, if deleting this edge would cause u and v to become disconnected. An $edge(u,v)$ in a network is a local bridge if u and v have no friends in common. If an $edge(u,v)$ with $W(u,v)$ functions as a bridge or a local bridge, $W(u,v)$ must be very small. As a result, after the thresholding step, the endpoints of a bridge are unlikely in the same community.

Next is the diffusion of membership. Similar with the diffusion of innovation, the cascade of joining community m can be model as a natural model of direct-benefit effects in networks[21]. According to the assumption of direct-benefit model, people will benefit from directly copying others' decisions in that the cost for the people within the same group is usually lower than that for people in different groups.

In our implementation, one node has only one community ID. If we relax this constraint by allowing every node to join as many communities as it needs, and the more communities it joins, the more taxes it pays. It's possible to generalize our algorithm to detect the overlapping communities.

B. Setting Of Threshold

A good choice of initial points, i.e. the core members, is very critical to ensure a good diffusion of the membership of different communities. On one hand, a good threshold Th has to make sure that at the current scale every real community has at least one core member, so Th must be not more than the smallest weight of different communities W_u. On the other hand, the core members of different communities should not be connected by edge with $W(u,v) \geq Th$(where $C(u) = i, C(v) = j, i \neq j$). So Th should be larger than the maximum weight W_l across different communities.

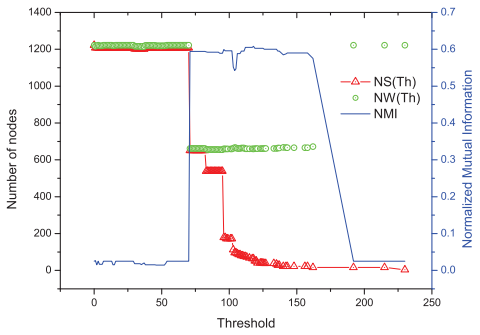
In fact we only need to find a $Th \in (W_l, W_u]$. The lower bound W_l and the upper bound W_u are the intrinsic properties of the social networks. The performance of our algorithm mainly depends on whether $Th \in (W_l, W_u]$ instead of the exact value of Th.

The information about the choice of threshold is still not known in advance. In order to find a rule to help us determine the threshold, we use the normalized mutual information to measure the quality of different partitions and plot the relationship between the NMI and other variables. We denote the number of members and strong members in community C_i as $NW_{C_i}(Th)$ and $NS_{C_i}(Th)$, when the threshold is Th. Experiments suggest that the best choice of Th has a relationship with $NW(Th)$, where $NW(Th) = \max\{NW_{C_i}(Th), i = 1, ..., m\}$. Fig. 3 shows the results on two typical benchmark networks, one having two communities and the other having many communities(where $NS(Th) = \max\{NS_{C_i}(Th), i = 1, ..., m\}$). Empirically, the NMI get its maximum value near the bottom of the $NW(Th)$ curve. And the maximum NMI usually corresponds to several thresholds.

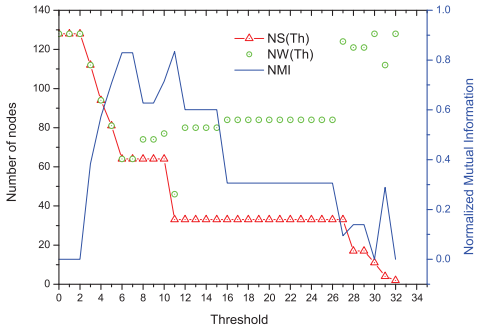
The value of Th should not be too low, or almost every node will belong to $V(Th)$. Yet Th should not be set too high either, or few nodes will belong to $V(Th)$ and few components will be found in $G(Th)$. When $NW(Th)$ reaches its minimum, the corresponding Th belongs to the right interval. The information about the number of communities found by our algorithm is also contained in the value of $NW(Th)$. So in our implementation, we choose $NW(Th)$ as our objective function, and test on different Th to get the minimum point of $NW(Th)$. In our implementation, a one-dimensional search procedure similar to the golden section search is used to avoid the unnecessary trials, which makes it possible to find the minimum value in a few trials.

C. Algorithm Complexity

We denote the average degree of the nodes by D_{avg}, and assume that the distribution of degree follows the power law. D_{avg} is related to the parameters of the power law distribution and the total number of nodes N.



(a) A sample network with 2 communities



(b) A sample network with 5 communities

Figure 3. Threshold Th vs NMI

The complexity of the thresholding step is $O(MD_{avg})$(M is the total number of edges), and that of the diffusion step is $O(ND_{avg})$. But since the threshold is not set down, and $\log W_{max}$ times of trials are needed, where the W_{max} is the largest weight in the network and in most case $\log W_{max}$ is less than 100. Finally the overall complexity of our algorithm is $O((M+N)D_{avg} \log W_{max})$.

V. Experimental Results

We test our algorithm on both classical social networks and synthetic networks. We adopt the NMI as the quality measure, and we use the source code in [3] to calculate the NMI. We compare our method with Infomap and Louvain method. According to the results in [20], Louvain method is one of the most efficient algorithms, and Infomap is acclaimed as the best method.

A. Tests On Real Networks

The first benchmark graph we test on is Zachary's karate club network[22]. It is a network of friendships between 34 members of a karate club and becomes the benchmark for all community detection algorithms. The second one is the network of American college football teams[23], with nodes standing for 115 football teams, and a link between two nodes representing they have played against each other. These teams

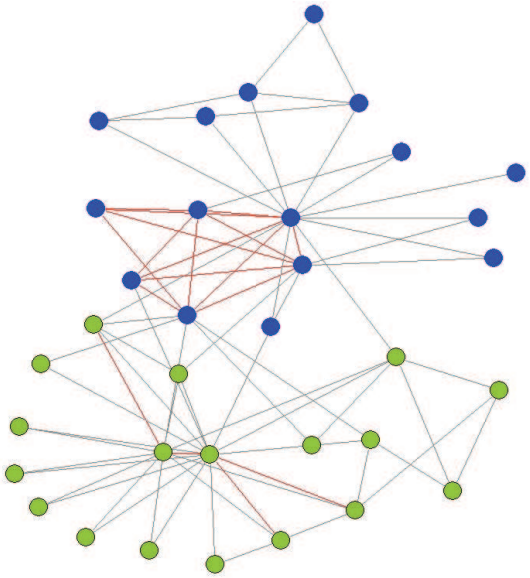


Figure 4. Our partition of the karate club network

TABLE I
THE NMI RESULTS ON REAL NETWORKS

	Karate	Football	Political blog
Louvain method	0.3604	0.6611	0.3701
Our algorithm	**0.8372**	0.6810	**0.6423**
Infomap	0.4593	**0.7930**	0.2961

are divided into 12 conferences. The third one is the political blog network[24] with over one thousand nodes. Every node represents a blog compiled around the time of the 2004 United States presidential election. These blogs can be divided into two parts, the liberal community and the conservative one. The results of our partitions are showed in Fig. 4-6. Different communities are marked by different colors.

The performances of three algorithms on the above mentioned networks are given in Table I. We compare the partitions of the algorithms with the real ones(The real partitions used in the calculation of NMI might be not the only meaningful partition). Our algorithm generally have a good division of different kind of networks. Both Louvain method and Infomap tend to yield more communities compared with the real communities structures. The karate club network and the political blog network each has two communities, and the football network has 12 communities. So our algorithm is better on karate club and political blog. We did think our algorithm might be intuitively more suitable for the football network in that teams in the same conference should have more common neighbors than teams in different conferences. However, from Table I, our performance in this ad-hoc network is worse than Infomap, which might be caused by some irregular nodes such as the independent teams.

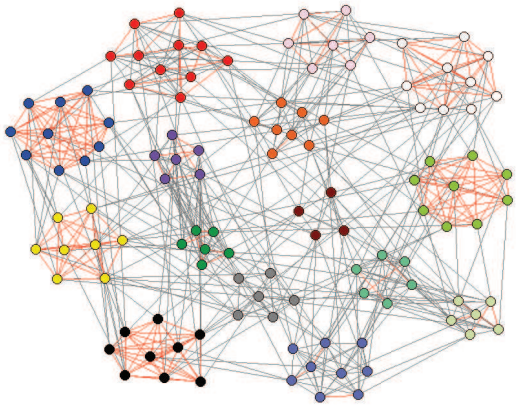


Figure 5. Our partition of the football network

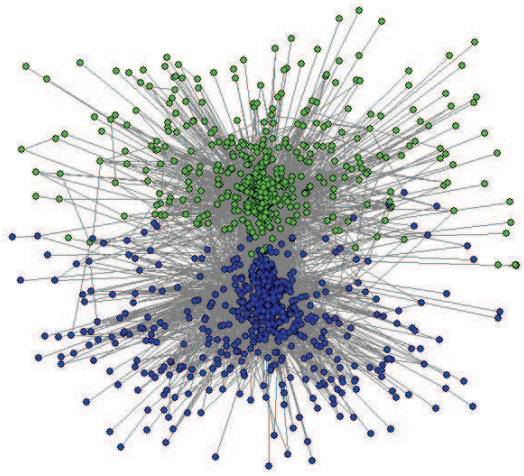
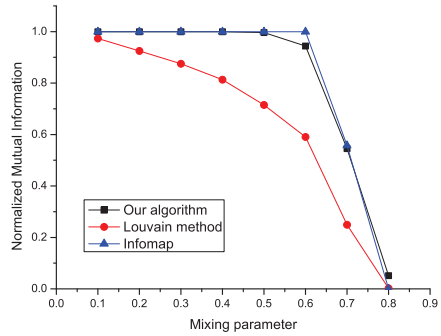


Figure 6. Our partition of the political blog network

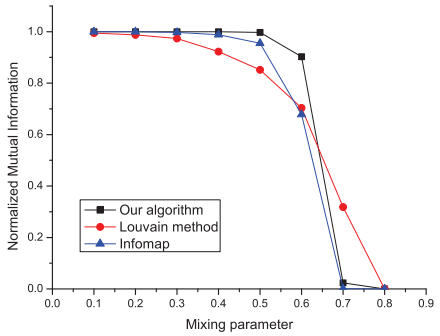
B. Tests On Synthetic Networks

Community structure information of social networks especially large social networks remains very scarce. So we test on some artificial networks. The GN benchmark is a well-known artificial network proposed by Girvan and Newman[23]. But the sizes of its communities are the same. The LFR benchmark was introduced in [25], which has generalized the GN benchmark by introducing power law distributions of degree and community size[20]. Since the degree and community size in many real networks are found to follow the power law, the artificial networks built by the LFR benchmark are more similar with the real ones. We generate the LFR benchmark by using the source code in [25]. More details about the LFR benchmark can be found in [25].

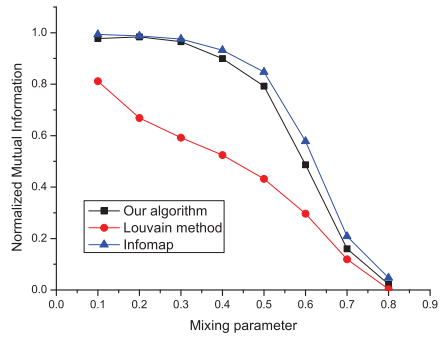
In Fig. 7 we show the performance on the LFR benchmark graphs. Some parameters of the LFR benchmark can be adjusted to generate different networks. The values of some parameters such as the number of nodes N, the average degree D_{avg}, the maximum degree D_{max}, the exponent of the degree



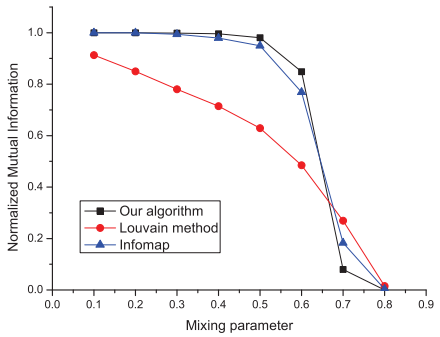
(a) N=1000, D_{avg}=20, D_{max}=50, γ=-2, β=-1



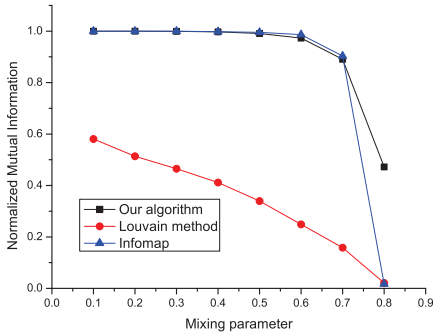
(b) N=1000, D_{avg}=30, D_{max}=50, γ=-2, β=-1



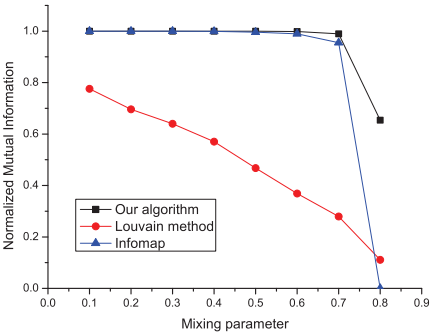
(c) N=1000, D_{avg}=20, D_{max}=150, γ=-1.5, β=-1.5



(d) N=1000, D_{avg}=30, D_{max}=150, γ=-1.5, β=-1.5



(e) N=5000, D_{avg}=30, D_{max}=150, γ=-1.5, β=-1.5



(f) N=5000, D_{avg}=40, D_{max}=150, γ=-1.5, β=-1.5

Figure 7. Tests of three algorithms on the LFR benchmark

distribution γ and the exponent of the community size distribution β are labeled under each chart. The mixing parameter μ of the benchmark is used to control ratio between the degree of intra-communities of a node and its total degree. As μ increases, there are more and more links bridging different communities, which makes the boundary of the community more ambiguous. In each test, we have averaged the value of the NMI over 100 realizations for each value of the mixing parameter μ.

Our algorithm has a steady performance on different networks and the quality is comparable with Infomap. In some situations our algorithm is even better. When D_{avg} increases, the performance of Infomap gets relatively worse, but our algorithm does not seem to be affected. The performance of Louvain method is not good, which might be caused by the well-known resolution limit problem of the modularity-based methods.

In order to compare the speed of three algorithms, we test on the LFR benchmark graphs with 50000 nodes($D_{avg} = 40$, $D_{max} = 200$, $\gamma = -1.5$, $\beta = -1.5$) and 100000

TABLE II
NMI AND RUNNING TIME ON THE LRF NETWORK WITH 50000 NODES

Mixing parameter μ	0.1	0.2	0.3	0.4
Louvain method	0.370/2.9s	0.3156/3.3s	0.3164/3.5s	0.203/3.6s
Our algorithm	1.0/9.9s	1.0/9.95s	1.0/10.69s	0.9988/10.5s
Infomap	1.0/10.8s	1.0/14.5s	1.0/18.0s	0.9994/20.4s
Mixing parameter μ	0.5	0.6	0.7	0.8
Louvain method	0.178/4.01s	0.139/4.18s	0.084/ 4.6s	0.001/7.67s
Our algorithm	0.998/10.7s	0.9928/12.2s	0.9784/ 12.1s	0.8544/17.5s
Infomap	0.999/21.7s	0.9987/27.6s	0.9967/ 63.1s	0.8995/84.5s

TABLE III
NMI AND RUNNING TIME ON THE LRF NETWORK WITH 100000 NODES

Mixing parameter μ	0.1	0.2	0.3	0.4
Louvain method	0.3817/7.96s	0.3199/8.5s	0.2764/9.34s	0.2428/10.2s
Our algorithm	0.9999/39.6s	0.9999/36.7s	0.9999/34.3s	1.0/33.4.s
Infomap	1.0/27.1s	1.0/37.0s	1.0/48.4s	0.9999/56.3s
Mixing parameter μ	0.5	0.6	0.7	0.8
Louvain method	0.1949/11.3s	0.1252/13.8s	0.047/16.7s	0.0119/16.7s
Our algorithm	0.9999/34.5s	0.9998/31.8s	0.9987/35.6s	0.9837/53.6s
Infomap	1.0/69.0s	0.9996/83.78s	0.9994/112.5s	0.9967/305.1s

nodes($D_{avg} = 50$, $D_{max} = 200$, $\gamma = -1.5$, $\beta = -1.5$). Both the NMI result and the running time they cost are given in Table II and III. Louvain method is the fastest, and in most cases our algorithm is faster than Infomap. As μ increases, more computational time is needed for all algorithms. Especially, it takes exponentially-growing time for Infomap to get a proper partition. When μ is small, the computational time for our algorithm and Infomap is similar. But when μ is large enough, for instances $\mu = 0.8$, our algorithm is much faster than Infomap. The implementation of our algorithm has not taken advantage of the natural parallelism of our algorithm. For example we can test on different thresholds at the same time. There is still lots of potential to improve our implementation.

VI. CONCLUSIONS

We have presented a hierarchical diffusion algorithm to detect the community structure in social networks. We use the local structure information of the network, the number of common neighbors, as the similarity measure. The social network we consider so far is unweighted and undirected, but it is possible to generalize our method to the weighted network by modifying the similarity measure properly. The similarity measure can be replaced by combination of two person's common interests, common friends, etc., which could enable it to be applied in the social networks with assorted attributes. Extension to the detection of overlapping communities is also possible by modifying the diffusion step of our algorithm. We are working on applying our algorithm on real social networks. We are working on this and try to test our algorithm on the online social networks with meaningful communities.

In some cases no partition will be found by our algorithm. This enables our hierarchical framework, since it can distinguish between pseudo-communities and meaningful ones.

ACKNOWLEDGMENT

This work is supported by National Basic Research Program of China(2010CB731401 and 2010CB731406).

REFERENCES

[1] L. Kuncheva and S. Hadjitodorov, "Using diversity in cluster ensembles," in *2004 IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, 2004.

[2] L. Danon, J. Duch, A. Arenas, and A. Dłaz-guilera, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 9008, p. 09008, 2005.

[3] A. Lancichinetti, S. Fortunato, and J. Kertsz, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, vol. 11, p. 033015, 2009.

[4] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, p. 26113, 2004.

[5] M. A. Porter, J.-P. Onnela, and P. J. Mucha, "Communities in networks," *Notices of the American Mathematical Society*, vol. 56, p. 1082, 2009.

[6] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*. Cambridge Univ Pr, 1994.

[7] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 2, pp. 172–188, 2008.

[8] A. Clauset, M. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, p. 66111, 2004.

[9] M. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E*, vol. 74, no. 3, p. 36104, 2006.

[10] R. Guimera and L. Amaral, "Functional cartography of complex metabolic networks," *Nature*, vol. 433, no. 7028, p. 895, 2005.

[11] J. Duch and A. Arenas, "Community detection in complex networks using extremal optimization," *Physical Review E*, vol. 72, no. 2, p. 27104, 2005.

[12] S. Fortunato and M. Barthlemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, vol. 104, no. 1, p. 36, 2007.

[13] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, p. P10008, 2008.

[14] P. Ronhovde and Z. Nussinov, "Multiresolution community detection for megascale networks by information-based replica correlations," *Physical Review E*, vol. 80, no. 1, p. 16109, 2009.

[15] M. Rosvall and C. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, p. 1118, 2008.

[16] E. Ravasz, A. Somera, D. Mongru, Z. Oltvai, and A. Barabási, "Hierarchical organization of modularity in metabolic networks," *Science*, vol. 297, no. 5586, p. 1551, 2002.

[17] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the National Academy of Sciences*, vol. 101, no. 9, p. 2658, 2004.
[18] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multi-scale complexity in networks," *arXiv*, vol. 0903.3178, 2009.
[19] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, p. 75, 2010.
[20] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis," *Phys. Rev. E*, vol. 80, no. 5, p. 056117, Nov 2009.
[21] S. Morris, "Contagion," *The Review of Economic Studies*, vol. 67, no. 1, pp. 57–78, 2000.
[22] W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, no. 4, pp. 452–473, 1977.
[23] M. Girvan and M. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, p. 7821, 2002.
[24] L. Adamic and N. Glance, "The political blogosphere and the 2004 us election: divided they blog." ACM New York, NY, USA, 2005, pp. 36–43, proceedings of the 3rd international workshop on Link discovery.
[25] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E*, vol. 78, no. 4, p. 46110, 2008.