Enhanced modularity-based community detection by random walk network preprocessing

Darong Lai* and Hongtao Lu
Department of Computer Science and Engineering, Shanghai Jiao Tong University, 800 Dong Chuan Road, 200240 Shanghai, China

Christine Nardini
Key Laboratory of Computational Biology, CAS-MPG Partner Institute for Computational Biology, Chinese Academy of Sciences, 320 Yue Yang Road, 200031 Shanghai, China

(Received 19 January 2010; revised manuscript received 21 April 2010; published 23 June 2010)

The representation of real systems with network models is becoming increasingly common and critical to both capture and simplify systems' complexity, notably, via the partitioning of networks into communities. In this respect, the definition of *modularity*, a common and broadly used quality measure for networks partitioning, has induced a surge of efficient modularity-based community detection algorithms. However, recently, the optimization of modularity has been found to show a resolution limit, which reduces its effectiveness and range of applications. Therefore, one recent trend in this area of research has been related to the definition of novel quality functions, alternative to modularity. In this paper, however, instead of laying aside the important body of knowledge developed so far for modularity-based algorithms, we propose to use a strategy to preprocess networks before feeding them into modularity-based algorithms. This approach is based on the observation that dynamic processes triggered on vertices in the same community possess similar behavior patterns but dissimilar on vertices in different communities. Validations on real-world and synthetic networks demonstrate that network preprocessing can enhance the modularity-based community detection algorithms to find more natural clusters and effectively alleviates the problem of resolution limit.

DOI: 10.1103/PhysRevE.81.066118 PACS number(s): 89.75.Hc

I. INTRODUCTION

Many real complex systems have recently been conveniently modeled as networks, with elements as vertices and relations between pairs of vertices as edges. Although as diverse as real complex systems can be, these networks exhibit number of interesting and important common properties, such as power-law degree distribution, small-world average shortest paths and high local interconnections, i.e., high clustering coefficients [1–3]. Recent researches reveal that networked systems are organized as interconnected subgroups of vertices, called communities. For example, communities may be sets of web pages with common topics [4], functional modules [5,6] and groups of peoples in different organizations [5].

Communities are in general subgroups of vertices with more chances to interconnect in the same subgroup but less often to connect to other vertices in different subgroups. Since the seminal work of community structure analysis in complex networks by Girvan and Newman [5], this topic has attracted growing attention. Modularity, an important and popular quality function, is then proposed to qualify the community decomposition [7], which greatly drives the advance of the researches on community detection. Modularity is often calculated based on the comparison of the actual network with its null model (or configuration model) [8]. The null model is a network with the same number of vertices and edges incident to each vertex, but randomly rewired. The larger the value of the modularity of a partition, the more exactly the network can be decomposed into its real communities. Modularity was first used in a class of divisive algorithms to select the partition with the highest value [7], and has frequently been used as a quality index ever since. Works of this type include spectral methods based on Laplacian matrix [9], information based divisive algorithm [10] and more. Modularity has also been used as a target function to be optimized directly, in algorithms such as simulated annealing [11], extremal optimization [12], spectral optimization [8,13,14] (for a more complete description see review [15]).

Although modularity is important and popular, Fortunato and Barthélemy recently found that modularity optimization has a resolution limit [16], indicating that clusters that are small with respect to the size of the whole network tend to be merged into larger ones by modularity optimization, even if they are cliques. Several works have tried to address this problem. For example, some works concentrate on recursively maximizing modularity for each single cluster found by previous partitions [16,17], without warranty of finding the proper communities; others have tried to define novel quality indexes as alternatives, such as modularity density [18], which are completely different from modularity-based ones and force to abandon the large body of previously developed efficient modularity-based methods. A spin model-based formulation with a tunable parameter has also been proposed [19,20]. In fact, the spin model-based method is designed not to address the resolution limit of modularity but to generalize modularity in view of statistical mechanics in physics. It is thus expected that such a generalized quality function (Hamiltonian) has resolution limit as well [20]. A parameter γ is used to tune modular resolution to reveal underlying hierarchical structure in networks if a prior information on the community structure is available. But in many cases the community structure is not known in advance, thus

*darong.lai@gmail.com

there is no simple way to find which γ will give the most probable mesoscale description of the network. As it has already been recognized [20], when the size distribution of communities of a network is broad (for example, collaboration networks), there is not unique proper γ to identify the most probable communities. Besides the aforementioned efforts, little attention has been drawn to preprocess the networks before they are fed into modularity-based algorithms. Arenas *et al.* [21] worked in this direction and proposed a method to modify the topology of the original network by adding self-loops with an identical weight r to every vertex and then used modularity optimization to detect dozens of possible candidates of community partitions. Such a method needs to sample a great number of values of r, and it is usually difficult to choose one or more probable partitions from so many possible candidates if *a priori* information is lacking (for larger networks this becomes increasingly difficult since partitions with equal number of groups are not necessary identical). In this paper, we similarly avoid developing a completely different community detection algorithm, but we aim at combining modularity-based methods with a network preprocessing strategy to analyze network communities. We will show that modularity-based methods coupled with network preprocessing can alleviate the problem of resolution limit in an effective way. Several other works use a similar strategy to attack multiscale modular structure or overlapping communities of networks [22,23], with the help of the concept of stability of a network [24]. However, these methods are fundamentally different from the one proposed in this paper with respect to the problems addressed and the underlying methodology (more thorough discussions of these differences will be given later when the proposed method is detailed).

Community structure reflects that each community in a network is to some extent independent from other communities. If a dynamic process as random walk (a process that a walker randomly takes from one vertex to one of its neighbors with a probability proportional to the weight of the edge connecting them) is triggered on one vertex, the process is more likely to persist on the vertices in the same community and far less on the vertices in different communities. Based on this fact, random walkers starting from vertices in the same community are likely to have similar patterns when visiting or influencing the whole network. In other words, random walkers starting from vertices in the same community will have similar trajectories when they randomly walk on the network, and will be likely to have very different trajectories from those of random walkers starting from vertices in different communities. If a suitable measure to capture such type of relations is available, we can differentiate edges by the partial and incomplete information on the community membership of their connected vertices, and devise a strategy to sharpen the differences.

In this paper, we indeed use random walk as a surrogate of the dynamic processes on the network to unveil such relationship. If two random walkers starting from two different vertices have highly similar patterns, the vertices are said to be in the same community with high probability, but if the patterns are very different, the vertices are in the same community with very low probability. As a result, we can set high positive weight on an edge to show that its two connected vertices are in the same community with high probability, and very low-positive weight otherwise. Obviously, the similarity of the patterns associated with any two vertices can be used as a way to increase or decrease the weight on the edge joining the vertices. After reweighting the edges on the original network, the topology of the network appears to be changed, as caused by our diminishing uncertainty on the community membership of the vertices on the network. By iteratively operating the same strategy on the newly weighted network, i.e., continuing to reduce the uncertainty, we can sharpen the differences of weights between intra- and intercommunity edges. After a few rounds of edge reweighting, the community structure of the original network becomes more obvious since the weights of the intercommunity edges nearly vanish. Interestingly, we can show that the cost of the network preprocessing is low, while the ability of the modularity-based community detection algorithms to find more natural communities in networks is enhanced.

In the following, after briefly introducing the concept of modularity and its resolution limit, in Sec. II, we extend the concept on binary networks to weighted ones, a step necessary to apply weighting concepts to modularity, originally defined on binary networks. We then propose our combined strategy to find communities in networks in Sec. III. We also give in Sec. IV a series of examples of application of the proposed method to real and synthetic networks with known community structure, to show its effectiveness. We further discuss the proposed method and draw conclusions in Sec. V.

II. MODULARITY AND ITS RESOLUTION LIMIT

Modularity guides us to identify the best decompositions of networks into communities. In this section, we first introduce the modularity for binary networks and its resolution limit when it is optimized, and then extend the analysis to weighted networks.

A. Binary networks

An undirected binary network is usually associated with a symmetric adjacency matrix A, whose element $a_{ij}=1$ if there exists an edge between vertex i and j, and $a_{ij}=0$ otherwise. Vertex degree k_i is the total number of edges incident to vertex i. Given a partition of an undirected network into C groups, modularity is defined as [7,8]:

$$Q = \sum_{s=1}^{C} \frac{l_s}{L} - \left(\frac{d_s}{2L}\right)^2 = \frac{1}{2L}\sum_{ij}\left(a_{ij} - \frac{k_i k_j}{2L}\right)\delta(c_i,c_j), \qquad (1)$$

where L is the total number of edges in the network and the sum is over C groups which can be cast into element-wise sum by function $\delta(\cdot,\cdot)$ [$\delta(c_i,c_j)=1$ if i and j are in the same community, i.e., community index $c_i=c_j$, and 0 otherwise]; l_s is the number of edges within the group s, and the total degree of vertices in this group is d_s. The goal of modularity-based community detection algorithms is to maximize the modularity. Modularity maximization sets an intrinsic scale to the communities that can be found, which means that communities that are smaller than this scale may not be dis-

covered [16]. This resolution limit depends on the total number of edges in the network and the degree of interconnectedness between pairs of communities, which can be as large as the order of size of the network. Fortunato and Barthélemy defined a sub-graph to be a valid community if in a binary network it satisfies:

$$\frac{l_s}{L} - \left(\frac{d_s}{2L}\right)^2 > 0. \tag{2}$$

To illustrate the behavior of modularity maximization, we briefly present an adaptation of the example used by Fortunato and Barthélemy to illustrate the concept [16]. Consider a binary network with L edges and with at least three communities (community C_1, C_2 and the rest of the network R_{net} that contains one or more communities). All these communities satisfy the definition in Eq. (2). Two different partitions of this network are considered. The first partition A considers C_1 and C_2 as independent communities, while the second partition B merges them into one larger community. The partition of the rest of the network R_{net} with respect to C_1 and C_2 is the same in both partition A and B. Intuitively, it is more reasonable to say partition A is more natural than partition B since both subgraphs C_1 and C_2 are communities under the definition given in Eq. (2). Consequently, the modularity Q_A of partition A should be larger than the modularity Q_B of partition B, i.e., $Q_A > Q_B$, which implies:

$$l_2 > \frac{2La_1}{(a_1+b_1+2)(a_2+b_2+2)}. \tag{3}$$

Here, l_2 is the number of edges in C_2; a_1 is the ratio of the number $l_{C_1C_2}$ of edges between C_1 and C_2 to the number of edges l_1 in C_1, and a_2 is the one between $l_{C_1C_2}$ and l_2. Similarly, b_1 is the ratio of the number of edges between C_1 and R_{net} to l_1, and b_2 is the ratio of the number of edges between C_2 and R_{net} to l_2. If there is no edge between C_1 and C_2, i.e., $a_1=a_2=0$, we always obtain partition A as we maximize modularity. In an undirected binary network, however, there must exist at least one edge between C_1 and C_2 if they are directly connected. Under the simplest case where $l_1=l_2=l$, the minimal value of the coefficients is $a_1=a_2=b_1=b_2=\frac{1}{l}$. In this case, if $l_2 \leq \sqrt{\frac{L}{2}}-1$, we obtain partition B as optimal. This analysis demonstrates that modularity sets a resolution scale to the communities found and some of them may be the combinations of other smaller communities below this scale. However, the network should be decomposed into natural groups, with more edges within groups but far fewer between them. If a binary network can be transformed into a weighted one by incorporating information on the structure of the network, a possible solution to this issue would be to add more weights inside the groups by putting fewer weights on the intercommunity edges, iteratively increasing the differences between inter- and intracommunity edges. The idea of random walk based iterative edge reweighting has been used in Markov Cluster Algorithm [25] and random walk based hierarchical clustering [26], however, to the best of our knowledge, its implications in network modularity were not foreseen. Weighting edges by random walk has also been found to optimize modularity [22,23], with the help of the concept of stability of a network [24]. Although both stability weighting and our method determine edge weights by random walks within a certain time scale (called *random walk length* in the current paper), the stability weighting strategy consists of a single time weighting and is fundamentally different from the one used in the current paper. Stability weighting, in fact, tends to add new edges to or remove old edges from the original network by increasing or decreasing the time scale, which modifies the connectedness of a network while our method does not, since it only weights differently already existing edges according to the knowledge on inter- and intracommunity edges. More importantly, stability weighting uses directly the random walk probabilities as similarities, which biases the weighting process since it gives higher values between high degree vertices and other vertices. Conversely, our approach uses the sum of probabilities as dimensional properties of vertex vectors and thus the similarities are induced from the similarities between vertex vectors (see Sec. III for more details).

B. Weighted networks

To adapt the concept of iterative reweighting to edges, a first necessary step consists of extending the modularity concept from binary networks to weighted ones. This can be easily done since weighted networks can be regarded as binary networks with multiple edges between pairs of vertices, if suitable weight unit is chosen [27]. A weighted network is determined by its weighted adjacency matrix W, whose element w_{ij} is a positive real number if there exists an edge between vertex i and j, and 0 otherwise. The weighted extension of modularity is [28]:

$$Q^w = \sum_{s=1}^{C} \frac{w_{ss}}{M} - \left(\frac{w_s}{2M}\right)^2 = \frac{1}{2M}\sum_{ij}\left(w_{ij} - \frac{w_{i\cdot}w_{j\cdot}}{2M}\right)\delta(c_i,c_j), \tag{4}$$

here M is the total weight of edges in the network; $w_{i\cdot}$ and w_s are respectively the weighted degree of vertex i and group s, while w_{ss} is the total weight of edges within group s. Similarly, the extension of the valid definition of community in a weighted network is:

$$\frac{w_{ss}}{M} - \left(\frac{w_s}{2M}\right)^2 > 0. \tag{5}$$

Since in an undirected binary network $a_1=a_2=b_1=b_2=\frac{1}{l}$ is the minimal value that we can obtain, nothing can be done to improve the resolution scale. However, if the binary network is transformed into a weighted one in an effective way, the minimal value of these coefficients can be as small as desired, since we can weigh the inter-community edges. Moreover, if the cost of network preprocessing is not excessively high we are sure to preprocess the networks. The benefit is that we can use already developed algorithms without any modification and we do not add important extra costs to achieve better results. With this in mind, we first consider a simple extreme case and assume $a_1^w=a_2^w=b_1^w=b_2^w=\frac{w_{min}}{w}$ with the total weights of C_1 and C_2 in a weighted network being

w_1 and w_2, and set $w_1=w_2=w$. As a result, we obtain partition A as our optimal partition when maximizing the modularity if:

$$w_2 = w > \frac{2Ma_1^w}{(a_1^w + b_1^w + 2)[(a_2^w + b_2^w + 2)} = \frac{2M\frac{w_{min}}{w}}{\left(\frac{w_{min}}{w} + \frac{w_{min}}{w} + 2\right)\left(\frac{w_{min}}{w} + \frac{w_{min}}{w} + 2\right)}. \quad (6)$$

Conversely, if $w \leq \sqrt{\frac{Mw_{min}}{2}} - w_{min}$, we get partition B as optimal by maximizing the modularity of the weighted network. Ideally, if we can put infinitesimal weights on the intercommunity edges, i.e., $w_{min} \to 0$, we can overcome the resolution limit problem imposed by modularity. That is to say, we will always find the more natural partition A of the network into communities.

III. RANDOM WALK NETWORK PREPROCESSING

Dynamic processes such as random walk on a network can explore the underlying network structure and this concept has already been employed to reveal communities in networks [22–24,29–33]. Starting from a vertex, a walker will randomly move to one of the neighbors of that vertex in the next time step with a probability determined by a transition matrix P. The element p_{ij} of the transition matrix is the ratio between the weight of edge (i,j) and the total weight of edges associated to vertex i, i.e., $p_{ij}=\frac{a_{ij}}{k_i}$ or $p_{ij}=\frac{w_{ij}}{w_i}$. In recent researches, it has been found that a random walker will be trapped for a longer time within than between communities [22,24,32]. In other words, random walkers starting from vertices in the same community will behave similarly when they randomly walk across the network. In the following, for convenience, we use *vertex behavior* to indicate the trajectory of a random walker starting from it. Based on this, if two vertices have very similar behaviors, we can say with high confidence that-if they are connected—these two vertices are in the same community, and thus the edge connecting them is very likely to be an intracommunity edge. Conversely, large differences between behaviors of two connected vertices mean that the edge they are connected with is very likely to be an intercommunity edge.

In order to get some sense of this approach, we briefly introduce an example making use of a social network (Zachary Karate Club Network [34]), which will be discussed in more details later in Sec. IV to compare the behaviors of vertices in the same versus different communities. In the real community partition of this network, vertex 2 and 8 are in the same community, while vertex 1 and 34 are in different ones. As expected, we can see from Fig. 1(a) that the behavior patterns of vertex 2 and 8 are comparably consistent, indicating that vertex 2 and 8 are in the same community with high probability. The far difference between the behavior patterns of vertex 2 and 34 gives the opposite evidence that they are unlikely to be in the same community [see Fig. 1(b)].

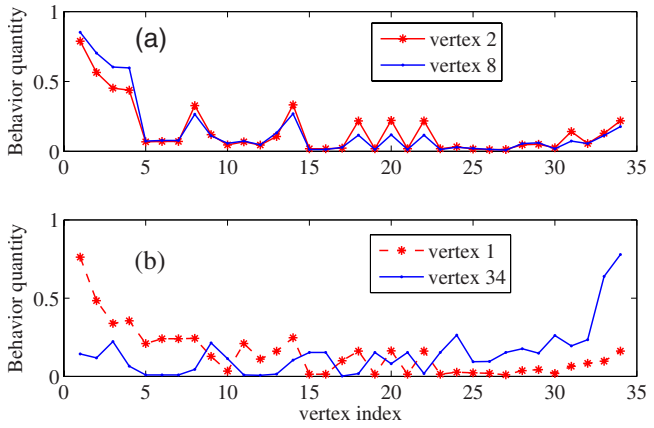

FIG. 1. (Color online) Illustration of random walk behavior patterns of vertices in Zachary karate club network [34]. (a) Behavior patterns of vertex 2 and 8 that are in the same community; (b) Behavior patterns of vertex 1 and 34 that are in different communities.

One of the implications of community partition is to correctly differentiate intercommunity edges from intracommunity ones. Modularity optimization can be regarded as a way to add as many weights as possible into communities and reduce weights of inter-community edges. A pair of vertices can easily and frequently be visited by random walks triggered on them if they are in the same community. As a consequence, we can give high weight to an edge with its two end points having similar behaviors under random walk processes and low weight otherwise. Based on a suitable similarity metric that captures well the similarities of behaviors among vertices, we can reweigh the edges of the original network without deteriorating its underlying community structure.

The probability of a walker starting from one vertex to reach another in t-step random walk is determined by matrix P^t (t is called random walk length here). P^t records the trajectories of random walks and has been used to reveal community structure in networks [32], where each row of P^t is considered as a vector in n-dimensional Euclidean metric space to induce similarities between pairs of vertices in term of Euclidean distances between vectors, denoted *PL-Similarity* here. We similarly view each vertex in network space as a vector but use instead $\Sigma_{\tau=1}^{t} P^{\tau}$ to track the trajectories of random walks. As the behavior patterns can be quantified as n-dimensional vectors, several similarity/dissimilarity measures can capture their coherence, such as square Euclidean distance [32], cosine similarity or angular distance [9], and so on. In practice, we empirically find that cosine similarity or angular distance captures such similarity much more accurately, since the dimension of the vector n is very high. What the similarity needs to reflect is the difference between the directions of two vectors, for instance to what extent they are parallel, not the absolute distance to indicate how far apart they are.

Both PL-Similarity and the one proposed here are based on the fact that random walks triggered on vertices in the same community will have similar trajectories. However, PL-Similarity only considers a t-step random walk, and one

of the difficulties of PL-Similarity is its sensitive dependence to parts of the network far away from the target vertices considered each time. If the network is nearly bipartite, PL-Similarity will have fluctuations which result in an unstable measure. Conversely, the similarity used here takes into account t types of random walks whose steps vary from 1 to t. Such a similarity emphasizes the contributions from the vertices near the target vertices currently considered, since the identification of two target vertices as being in the same community is mainly dependent on the interconnectedness between the target vertices and the nearby vertices, not on that between the target vertices and the far away vertices. The cosine similarity of two vectors v_i and v_j is defined:

$$Sim_{cos}(v_i,v_j) = \frac{(v_i,v_j)}{\sqrt{(v_i,v_i)}\sqrt{(v_j,v_j)}}, \tag{7}$$

where (v_i,v_j) is the inner product of vector v_i and v_j. If two behaviorial vectors v_i and v_j are highly consistent, i.e., v_i almost parallels to v_j, cosine similarity $Sim_{cos} \to 1$; but if v_i and v_j are orthogonal, i.e., elements of v_i and v_j are alternatively 0, $Sim_{cos} \to 0$. Thus, Sim_{cos} can be used as the probability of an edge to lie in a community.

For each edge of the original network, we set as weight the cosine similarity of the behavior patterns of its two connected vertices. A new weighted network is then produced irrespectively from whether the original network is binary or weighted. Repeating the same strategy on the newly obtained weighted network, we iteratively sharpen the differences between weights of intra- and intercommunity edges. After the original network is preprocessed, we feed the finally weighted network into an already developed community detection algorithm and obtain the community decomposition of the original network.

In this paper, we focus on four main modularity-based community detection algorithms. Two have comparable accuracy and high efficiency, and are the eigenvector-based method proposed by Newman [14], denoted as *EigenMod*, and a very fast algorithm developed by Blondel *et al.* [35], denoted as *FastMod*. Two other stochastic algorithms are also considered in the cases of small networks in Sec. IV: one by Guiméra and Amaral [11] uses simulated annealing to optimize modularity, and is denoted *SAMod*, and the other by Duch and Arenas [12] uses extremal optimization, named *EOMod*. The two stochastic algorithms can obtain higher accuracy if they are run several times to select the best results, but consequently this requires much higher computation time than EigenMod and FastMod. The four modularity-based algorithms all have very good performance in terms of accuracy, and any improvement on the results obtained by these algorithms indicates that the proposed strategy is useful and effective.

We here briefly introduce the four modularity-based algorithms. EigenMod reformulates modularity in a quadratic form:

$$Q = \frac{1}{2M}\sum_{ij}\left(w_{ij} - \frac{w_{i\cdot}w_{j\cdot}}{2M}\right)\delta(c_i,c_j) = \frac{1}{4M}\sum_{ij} s_i B_{ij} s_j = s^T B s \tag{8}$$

and casts community detection as Eigen-decomposition of the matrix B. Here $B_{ij}=w_{ij}-\frac{w_{i\cdot}w_{j\cdot}}{2M}$ and s is the membership vector representing the partition of the network into two communities (vertex i is assigned to one community if $s_i=+1$, or to another one if $s_i=-1$). EigenMod searches vector s to be as much as possible parallel to vector u corresponding to the largest eigenvalue, which is achieved by extracting the signs of the components of the vector u: $s_i=+1$ if $u_i>0$ and $s_i=-1$ if $u_i\leq 0$. The result is further refined by vertex sweep to get the highest increase of modularity. This bisecting procedure continues on each of the clusters until modularity cannot be improved any further. Differently from EigenMod, FastMod first treats each vertex as a separate community and then computes the modularity gain by removing vertex i from its community and moving it to the community of its neighbor j. If there exists a moving with highest positive modularity gain, vertex i is put in that community and otherwise stays in its original community. This step of vertex moving is repeated until no improvement can be achieved. The next step builds a new network whose vertices are communities found in the first step, and the weights of edges between new vertices are the sum of weights of edges between vertices in the communities represented by these new vertices. The two steps are repeated until the local maximal value of modularity is obtained. FastMod is very fast and can deal with networks with millions of vertices, based on the fact that the modularity gain of vertex moving can be easily computed:

$$\delta Q = \left[\frac{\sum w_{in} + k_{i,in}}{2W} - \left(\frac{\sum w_{tot} + k_i}{2W}\right)^2\right] - \left[\frac{\sum w_{in}}{2W} - \left(\frac{\sum w_{tot}}{2W}\right)^2 - \frac{k_i}{2W}\right], \tag{9}$$

here w_{in} is the total weight of the edges in the community where vertex i can be moved to, and w_{tot} and $k_{i,in}$ are the total weight of the edges associated with the community and the degree of vertex i in that community, respectively. SAMod uses minus modularity as energy to be minimized. At each temperature, a number of random updates are accepted with some probability: if the energy is reduced the update is definitely accepted, otherwise it is accepted with a probability inversely proportional to the exponential increase of energy. The random updates are generated jointly by a series of individual vertex movements from one community to another and collective movements involving merging two communities and splitting a community. EOMod reformulates modularity as the sum of individual vertex contributions. It initially splits the whole network into two random parts with equal number of vertices, and each connected components is considered an initial community. It then iteratively moves the vertex with lower modularity contribution from one community to another until local maximal of modularity is reached. After that each community found is treated as a separate network by deleting its links to other communities and the process continues until the total modularity cannot be improved any further.

We can summarize the procedure to combine network preprocessing with already developed community detection algorithms as follows:

Step 1. Set the random walk length t to a suitable value, and calculate the quantity $\Sigma_{\tau=1}^{t}P^{\tau}(i)$ for each vertex i of the input network with a weighted adjacency matrix W, where $P^{\tau}=(D^{-1}W)^{\tau}$, and D is the diagonal matrix with weighted degrees of vertices on its diagonal;

Step 2. For each edge (i,j) of the network, calculate the cosine similarity of behavior patterns of vertex i and j, i.e., $w_{ij}=\cos[\Sigma_{\tau=1}^{t}P^{\tau}(i),\Sigma_{\tau=1}^{t}P^{\tau}(j)]$, and set w_{ij} as its new weight;

Step 3. Run Step 1 and Step 2 iteratively for several rounds I.

Step 4. Feed the finally weighted network into a community detection algorithm adopted, and output the community decomposition of the original network.

The procedure can be divided into two main phases: preprocessing of the original network (step 1–3) and decomposition of the newly obtained network into communities (step 4). The complexity of step 1 depends on the computation of $\Sigma_{\tau=1}^{t}P^{\tau}(i)$, which can be done on average in $O(\langle k\rangle^{t})$, where $\langle k\rangle$ is the average number of neighbors of vertices (degrees) in the network. If the degree of a network is bounded, as it is always the case in real applications, calculating $\Sigma_{\tau=1}^{t}P^{\tau}(i)$ is done in constant time. Thus the cost of step 1 is $O(n)$. It is easy to show that for L edges in the network the average cost of step 2 is $O(L\langle k\rangle)$, which is $O(n\langle k\rangle)$ for a sparse network. Consequently, the total cost of the first phase is $O(IL\langle k\rangle)$ or $O(In\langle k\rangle)$ for a sparse network. As we will show later in Sec. IV, few iterations of step 3 are sufficient to distinguish the intra- and intercommunity edges and thus the complexity of the first phase is $O(n)$ if the network is sparse and its degree is bounded. The cost of the second phase is that of the community detection algorithm adopted, which are $O(n^2\log n)$ for both EigenMod and EOMod, and approximately $O(n)$ for FastMod. The complexity of SAMod is far higher than that of the three others and cannot be simply estimated in terms of n or L for its dependence on the parameters used. In all applications in this paper, we find that the first phase of the procedure is fast and thus does not impose heavy burden on the total cost.

Before applying the procedure to detect communities in networks, the parameter t of random walk length needs to be specified in advance (step 1). Since the convergence speed of random walk is exponential, t in practice should not be chosen too large, i.e., possibly not greater than $\log(n)$. To examine the influence of t on the cosine similarity more thoroughly, we consider an extreme case when $t\to\infty$. As it has already been found [32,36], when $t\to\infty$ the probability of random walker being on a vertex j only depends on the degree $w_{j\cdot}$ of vertex j:

$$\lim_{t\to\infty}P_{ij}^{t}=\frac{w_{j\cdot}}{2M},\quad 1\le i\le n \tag{10}$$

Let t_c be the converged steps of random walk, then if $t\ge t_c$, $P^{t}\approx C$. Here C is a $n\times n$ constant matrix whose rows are identical vectors with j^{th} entry being $\frac{w_{j\cdot}}{2M}$. The behavior quantity $\Sigma_{\tau=1}^{t}P^{\tau}$ can thus be separated into two partial sums over $t_x<t_c$ and $t_\alpha\ge t_c$:

$$\sum_{\tau=1}^{t}P^{\tau}=\begin{cases}\sum_{\tau=1}^{t_x}P^{\tau}+(t_\alpha+1-t_c)C, & t\ge t_c,\\ & t_\alpha=t;\\ \sum_{\tau=1}^{t_x}P^{\tau}+0, & t<t_c,\\ & t_x=t.\end{cases} \tag{11}$$

Let $X=\Sigma_{\tau=1}^{t_x}P^{\tau}$ and $P_\alpha=\Sigma_{\tau=1}^{t}P^{\tau}-\Sigma_{\tau=1}^{t_x}P^{\tau}$. Cosine similarity can then be rewritten in a similar way:

$$Sim_{cos}(x_i,x_j)=\frac{(x_i,x_j)+(x_i,\alpha)+(x_j,\alpha)+(\alpha,\alpha)}{\sqrt{(x_i,x_i)+2(x_i,\alpha)+(\alpha,\alpha)}\sqrt{(x_j,x_j)+2(x_j,\alpha)+(\alpha,\alpha)}}. \tag{12}$$

here x_i and α are the i^{th} rows of X and P_α, respectively. When $t\to\infty$, $(\alpha,\alpha)\to\infty$, which makes $Sim_{cos}(x_i,x_j)\to 1$ for any pair of edge end points i and j. Edge weighting is in such case equivalent to simply ignoring all edge weights in the original network and thus produces a binary network. This is undesirable since we never obtain any information on the network structure but loose valuable information contained in edge weights. Information needed to distinguish inter- and intracommunity edges comes merely from the local structure of a network. Therefore we need to choose a small value of t to preprocess network by edge weighting. The strategy to choose the value of t is similar to that for choosing the number of principal components able to reveal the internal structure of the data with principal component analysis (PCA [37],) widely used in data mining and pattern recognition. If we imagine P^{τ}s $(1\le\tau\le t)$ like principal components, the value of $t<t_c$ is chosen to best explain local structure of the network and can be thought of as follows: the P^{τ}s corresponding to smaller values of τ account for as much of the information on local structure in the network as possible. It is also possible to choose $t<t_c$ as large as possible, however, random walks corresponding to larger value of t give little additional information on the network's local structure while introducing additional information on global network structure that is not so useful in distinguishing inter- and intracommunity edges and sometimes misleading (i.e., relevant information on local structure contains noises). Furthermore, larger t results in more iterations for the procedure to eliminate unnecessary or misleading information. Although there is so far no theoretical foundation for determining the optimal values of t, in our experience, we observed that in most cases small values like $t=2,3,4$ are sufficient to capture the information on local structure of the network without over determining it ($t=2,3,4$ are chosen to reflect the fact that triplets, triangles and rectangles are more frequently found within communities than between). Moreover, for a very sparse network the value of t should increase while for a very dense network decrease, due to the exponential converging speed of the random walk. The choice of small values of t achieves good empirical compromise between efficiency and accuracy as we will see in a series of examples in next section.

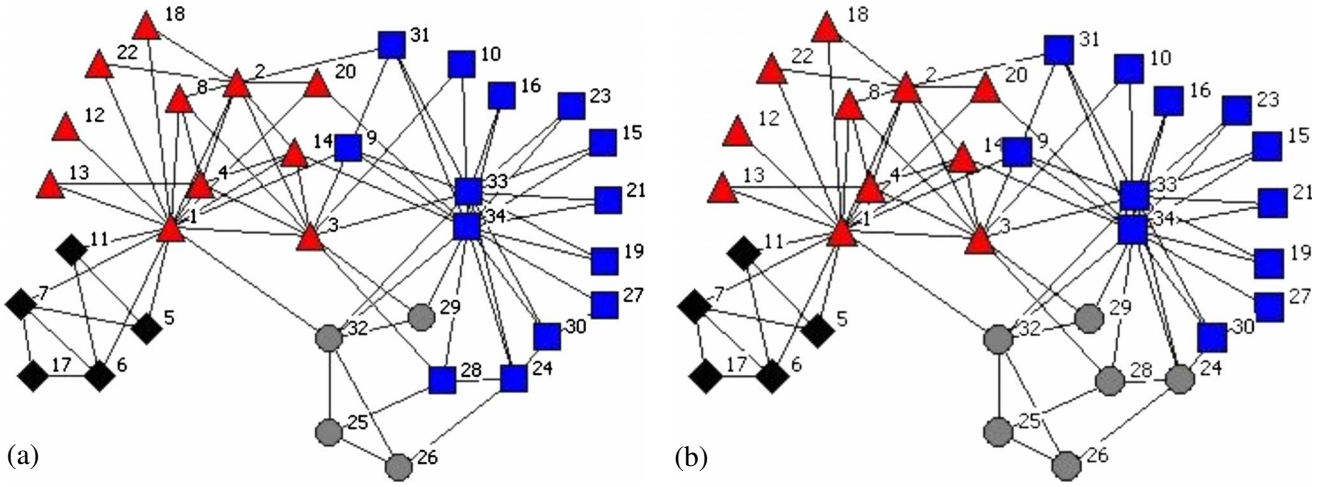

FIG. 2. (Color online) (a) Community partition of Zachary karate club network when $t=3$; (b) Community partition of Zachary karate club network when $t=4$. All four modularity-based algorithms produce the same results when optimizing the modularity of partitions of the reweighted network.

IV. APPLICATIONS

To experimentally validate the effectiveness of the combined procedure, we apply it to a number of real and synthetic networks with known community structure.

A. Zachary's karate club network

This network is constructed by W. W. Zachary [34], and consists of 34 vertices representing members of a karate club and 78 edges to show friendship relations between its members. The network had been split into two disjoined groups (the first one consists of 16 vertices: 1–8, 11–14, 17–18, 20, 22, and the second one consists of the rest) due to the disagreement between the administrator and the instructor of the club during the years in which W. W. Zachary studied it. The network has become one of the well-known benchmarks to test community detection algorithms [12–14,21,29], and, when optimizing modularity, it is always split into four groups [similar to the ones shown in Fig. 2(b) except vertex 10 is put in the red up-triangle group, with modularity value 0.4188 in most cases, including EigenMod and EOMod].

The output of FastMod somewhat depends on the input order of the vertices though the influence is limited. When we permute the input order of vertices, the maximal modularity that FastMod can achieve is 0.4198 (also the highest modularity that SAMod can achieve), and the exact partitioning is shown in Fig. 2(b) where vertex 10 is now in the group with most of its friends. To test the random walk preprocessing, we choose two random walk lengths: $t=3$ and 4. We iterated 5 times to reweigh the network ($I=5$), and fed the novel network into all four modularity-based algorithms introduced in Sec. III. Results are shown in Fig. 2 with different random walk lengths. For FastMod, we permuted the input order of the vertices and chose the partition with the highest modularity. For the two stochastic algorithms SAMod and EOMod, we ran them under no less 30 different initial conditions and selected the partitions with the highest modularity on the weighted network. The difference between the partitions in Figs. 2(a) and 2(b) consists of different subdivision of the second groups from the real splitting. Intuitively, the partition in Fig. 2(a) is more natural than the one in Fig. 2(b) since vertex 25, 26, and 32 form a triangle and the blue square vertices form a much more compact group. As we have seen, when optimizing modularity calculated on the original network, both EigenMod and EOMod obtained lower modularity (0.4188) while FastMod and SAMod can achieve higher modularity (0.4198). After the network was preprocessed, the results of all the four modularity-based algorithms became consistent. Such an interesting phenomenon indicates that modifying the topology of the original network by edge reweighting changes the searching space of modularity landscape and sometimes facilitates the heuristics to achieve optimal modularity.

B. Network of American college football teams

We applied the proposed procedure to the network of American college football games during the fall regular season 2000 [5]. The network contains 115 vertices representing teams and 613 edges representing games played between the teams. The 115 teams come from 12 conferences and games are more frequent between teams from the same conference than between teams from different conferences. We first applied all the four modularity-based algorithms to this network to reproduce the results. All the methods decompose the network into 10 communities, but the values of modularity obtained are different, namely 0.6009, 0.6046, 0.6044, and 0.6043 by EigenMod, FastMod, SAMod, and EOMod, respectively. The differences among these partitions are the different assignments of vertices 37 (Central Florida), 59 (Louisiana Tech), 60 (Louisiana Monroe), 64 (Middle Tennessee State), 83 (Notre Dame), and 98 (Louisiana Lafayette). For example, vertices 37, 59, 60, 64, and 98 are merged by EOMod into the conference "Southeastern," while by SAMod vertices 59, 60, 64, and 98 are put into "Conference USA" and vertex 37 is moved to conference "Mid-American." Although the difference between values of

modularity obtained by EigenMod and FastMod is a little more significant, the only difference between the partitions of EigenMod and FastMod is the classification of vertex 83. We thus conjecture that the searching space of values of modularity on the original network contains many local minima resulting in such inconsistent behaviors of the four modularity-based algorithms.

To preprocess this network, we again used two random walk lengths: $t=3$ and 4, and fed the processed weighted network into all four algorithms. Although these algorithms give different partitions when directly acting on the original network, the partitions obtained become consistent when the preprocessed one is fed. To see more clearly the decomposition behavior of the proposed procedure, we visualized the confusion matrix whose elements (i,j) represent the number of common vertices found both in community i by the proposed procedure and in real community j. The columns of the confusion matrix were rearranged to make the diagonal elements as large as possible to obtain the one-to-one correspondence between the communities found and the real ones, see Fig. 3. The procedure decomposes the network into 12 communities when $t=3$ and 11 communities when $t=4$. The difference is that community indexed 11 is merged into community indexed 7, i.e., the elements of the eleventh row of the matrix in Fig. 3(a) are moved and added to the seventh row [see the difference between the confusion matrices in Figs. 3(a) and 3(b)]. The group in green square and the blue arrow in Fig. 4 illustrate this merging. In these two partitions, the real community "Independence" (indexed 6) is disassembled and its members are distributed to three other conferences. This is due to the fact that teams in the conference "Independence" play more games with interconference teams than those in their own conference, which is against the definition used here that vertices in the same community have more edges than vertices in different communities. The values of modularity calculated on the original network of the partitions with $t=3$ and that with $t=4$ are 0.6032 and 0.6010, respectively. Although the values obtained by the proposed procedure appear to be smaller than the maximal one (i.e., 0.6046), we find that the partitions are more natural than those obtained by solely applying the detection algorithms to the network. If we use the number of correctly classified vertices as quality index, i.e., accuracy, we can quantify this assumption. To calculate the accuracy, we first searched the correspondence with maximal total sum of common members between the communities found by the procedure and the real ones. In order to make the correspondence to be a one-to-one mapping, we assigned the real community index to the newly found one with the largest common members and each real community index could only be assigned once. We continued this assignment until all found communities had been reindexed or the indexes of the real communities were used up. Consequently, the accuracy is the sum of the number of common members between found communities and their corresponding real communities [for example, the accuracy of the proposed procedure at $t=3$ is the sum of the diagonal elements of the matrix shown in Fig. 3(a)]. This type of accuracy is a very stringent measure to put emphasis not only on the correct classification of the vertices into communities but also on the correctness of the number of communities found. By this index of performance, both partitions found by the proposed procedure correctly classify 104 vertices out of 115 vertices (vertices in ellipses and square are considered to be misclassified, see Fig. 4), while partitions found by directly feeding the network into the four modularity-based algorithms correctly classify 100 vertices. The higher accuracy by the proposed procedure results from the different classification of the conference "Sun Belt." The conference "Sun Belt" is merged by pure modularity-based algorithms into the conference "Mountain West" (due to resolution limit), while most of its team members are correctly identified by the proposed combined procedure as an independent community. The advantages of random walk network preprocessing in this case are twofolds. First, network preprocessing makes the behaviors of the four modularity-based community detection algorithms consistent. Second, and more importantly, network preprocessing helps the four algorithms to find more natural communities, i.e., it cannot

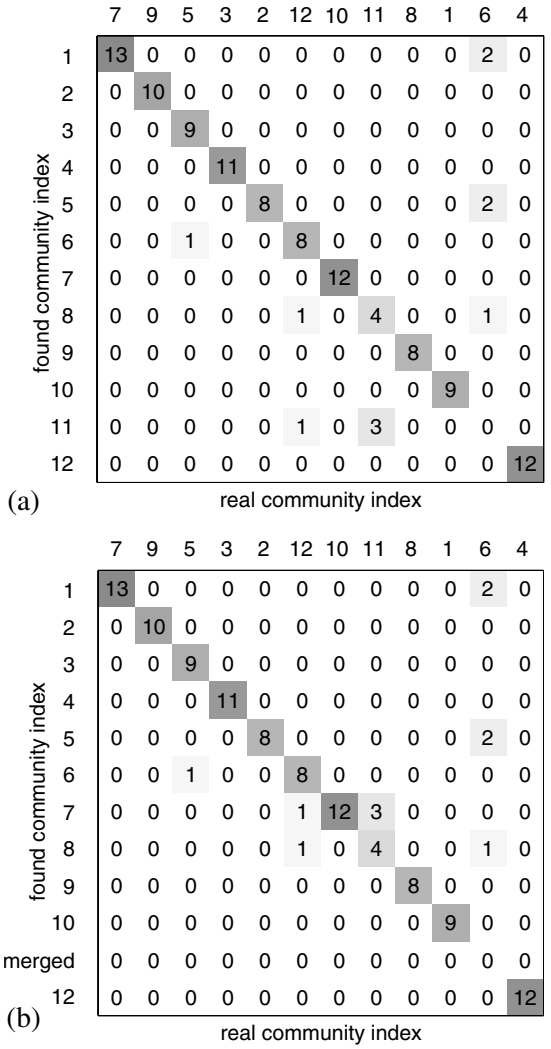

FIG. 3. Confusion matrix constructed by the communities found by the proposed procedure and the real ones. Row labels represent the labels of the communities found and column labels are real community indexes (as they are labeled in [5], plus 1). The columns are rearranged to make the diagonal elements as large as possible. (a) Confusion matrix with $t=3$. (b) Confusion matrix with $t=4$.

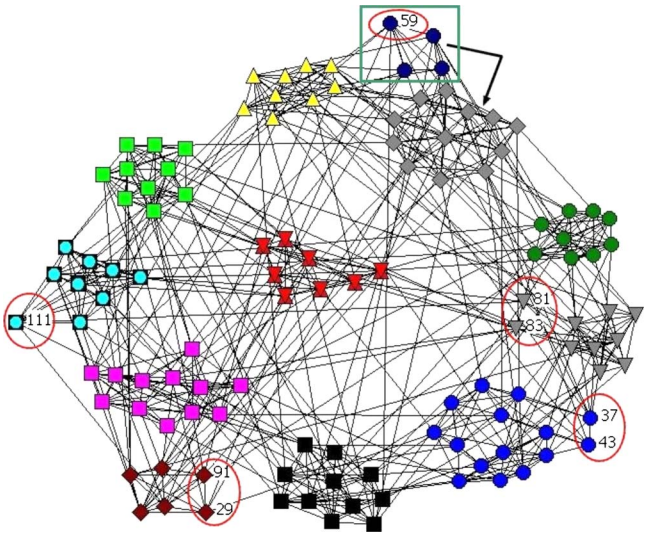

FIG. 4. (Color online) Result with random walk length $t=3$. In this case, the combined procedure finds 12 communities, which is consistent with the ground truth number of communities. 11 out of 115 vertices in ellipse and in square are misclassified by the suggested accuracy index.

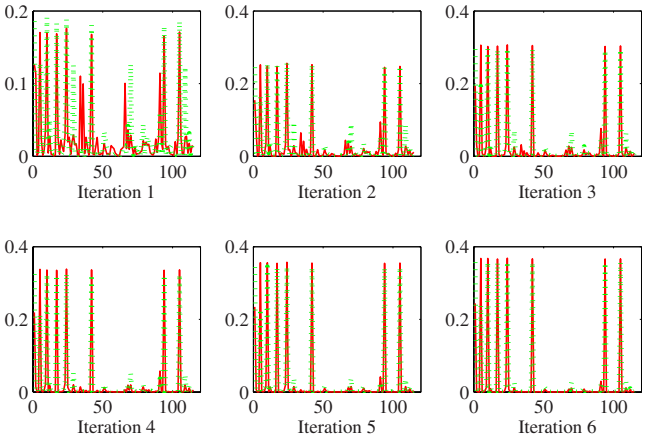

FIG. 5. (Color online) Iteration quickly makes the behavior patterns of vertices in the same community consistent. Vertex 1 (red solid line) and vertex 5 (green dot line) in the network of American college football teams are chosen to illustrate this phenomenon.

only assign vertices to their true communities but also predict the number of communities more correctly.

We also used this network to show that iterative edge reweighting makes the behavior patterns of vertices in the same community become consistent. For example, vertex 1 and vertex 5 in this network are in the same community. Figure 5 shows that their patterns quickly become consistent after 3 iterations. Iterative reweighting puts iteratively heavier weights on the intracommunity edges and lighter weights on the inter-community ones, which makes a random walker starting from a vertex spend most of its time visiting its neighbors in the same community and hardly reach vertices outside the community. In all the applications in the present paper we iteratively weighted each network 5 times ($I=5$) if not explicitly declared.

C. Clique chain networks

We tested our method on the same synthetic data set on which Fortunato and Barthélemy identified modularity resolution limit [16]. Figure 6 gives two types of clique chain networks. The network in Fig. 6(a) consists of 30 identical cliques, with 5 vertices each, connected by a single edge. The one in Fig. 6(b) is a network with two pairs of identical cliques and the big pair has 20 vertices each while the other pair is a pair of five-vertex cliques. Intuitively, the cliques are the natural communities and proper community detection algorithms should find them correctly. If the network is not preprocessed, none of the four modularity-based methods can correctly identify the regularity of cliques being communities. The communities they detect are the combinations of cliques. If the network is preprocessed first, i.e., iteratively reweighting edges with $t=3$ or $t=4$, they all correctly decompose the network into cliques. We further performed testing on ten similar clique chain networks by varying the number of cliques chained. The results are summarized in Table. I. Note that the combined procedure correctly detects the number of cliques over the whole range, while the four algorithms initially find the correct number of cliques but fail when the number of cliques increases as shown analytically [16]. Similarly, the four algorithms find 3 communities of the network in Fig. 6(b) with two 20-vertex cliques correctly detected and the other one identified as the combination of the five-vertex clique pair. In contrast, the combined procedure detects all 4 cliques correctly. The success of the combined procedure is due to the incorporation of the information on the local structure of the networks.

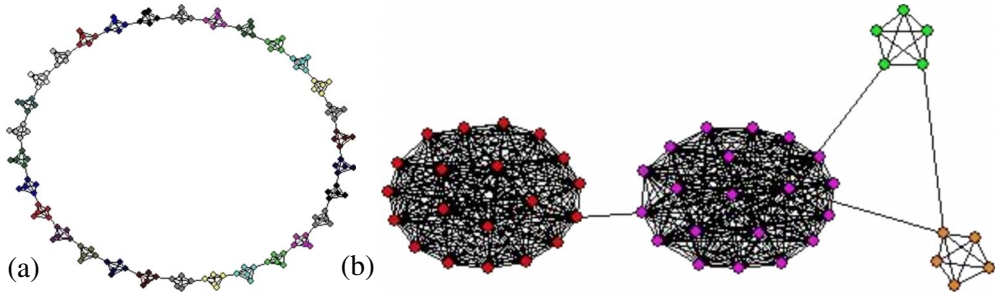

FIG. 6. (Color online) Illustration of two types of clique chain networks. (a) Network made of 30 cliques, with 5 vertices each, connected by a single edge. (b) Network made out of two pairs of identical cliques, with big pair having 20 vertices each and the other pair 5 vertices each.

TABLE I. Results for the resolution limit test on networks made out of cliques chained by a single edge. The number of cliques ranges from 20 to 30. Without network preprocessing, all four modularity-based methods cannot detect the correct number of cliques as it increases. If the networks are preprocessed as suggested in the paper, all of them detect the cliques correctly. Here EitherMod represents either of the four modularity-based algorithms.

	Number of cliques										
Methods	20	21	22	23	24	25	26	27	28	29	30
EigenMod only	20	21	14	15	16	16	16	16	16	16	16
FastMod only	20	21	22	12	12	13	13	14	14	15	15
SAMod only	20	21	21	12	12	13	14	14	15	16	16
EOMod only	20	21	14	14	12	13	15	15	16	16	16
Preprocessing+EitherMod	20	21	22	23	24	25	26	27	28	29	30

Besides the arguments in Sec. III, we further generated a series of the same type of networks as the one shown in Fig. 6(a) but varied the number of chained cliques from 100 to 2000 to get more intuition on the effect of parameters t and I on the behavior of the proposed procedure. Random walk lengths t were chosen from 1 to 6, and at each value of t the number of iterations I increased from 1 to the first value able to detect the correct number of cliques chained in a network. Table II summarizes the results. The information on local structure of such type of clique chain networks can be fully expressed by $t=1$ since all the vertices in the same community (clique) can be visited in a one step walk. Thus, the number of iterations necessary to extract relevant information is only one in all cases. When t increases, information on global structure of the network is also included, therefore more iterations are needed to extract relevant information on local structure from the one on global structure that may hinder the procedure to quickly differentiate inter- and intracommunity edges. Such type of clique chain networks are so special that they are different from most of real-world networks. When applying the proposed procedure to networks with realistic features, different combinations of the parameters t and I show very good and similar performance if networks possess clear community structure, as will see in the coming subsection.

D. LFR benchmark networks

All the networks we have tested so far are small or special. To test the random walk preprocessing on larger networks, we used a recently introduced class of benchmark networks with realistic features such as power-law distributions of degree and community size [38]. The vertex degrees of a network are controlled by a power law distribution with exponent t_1, and the community size also distributes according to power law with exponent t_2. The ratio between the external degree of each vertex with respect to its community and the total degree of the vertex is determined by a common *mixing parameter* μ. The larger the value μ of a network is, the harder it is to detect communities in it.

SAMod and EOMod are stochastic algorithms and thus need to rerun many times to obtain comparable results, this results in much higher computational cost. In addition, compared to SAMod and EOMod even with a large number of repetitions, EigenMod and FastMod are empirically found to perform much better on LFR networks both in accuracy and efficiency. For this reason in this validation on LFR networks we only reported the improvements on EigenMod and FastMod to see if the proposed procedure is effective. We first generated a set of undirected binary networks with 1000 vertices. For the exponent of the degree distribution and that of the community size, the default values provided by the algorithm were used: $t_1=-2$, $t_2=-1$. The mixing parameter μ varies from 0.05 to 0.8. The average degree and the maximal degree are 20 and 50, respectively. To preprocess these networks, we set the random walk length to 3. We used the same strategy to calculate accuracy and the results are presented in terms of the fraction of correctly classified vertices to ease figure readability. The performance comparison on

TABLE II. Relationship between random walk length t and number of iterations I to identify correct number of cliques in clique chain networks. Each clique contains 5 vertices and the number of cliques varies from 100 to 2000.

	Number of cliques/100 I												
t	1	2	3	4	5	6	7	8	9	10–13	14–16	17	18–20
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	3	3	4	4	4	4	5	5	5	5	6	6
3	2	3	4	4	5	5	5	5	6	6	6	6	7
4	3	4	5	5	5	6	6	6	6	7	7	7	8
5	3	4	5	6	6	7	7	7	7	8	8	9	9
6	3	5	6	7	7	7	8	8	8	9	10	10	10

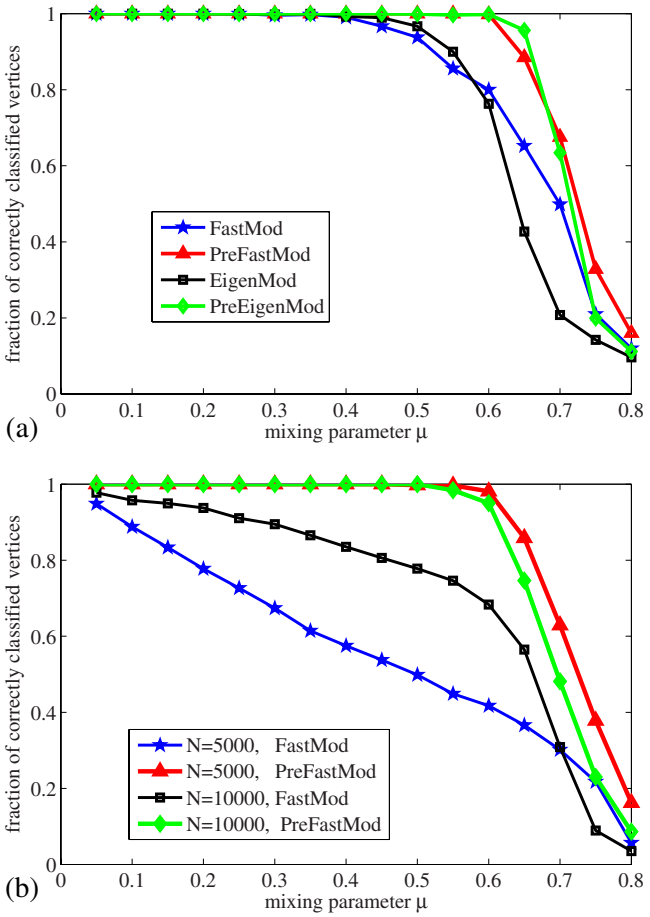

FIG. 7. (Color online) Performance comparison with or without random walk network preprocessing. (a) Performance on networks with 1000 vertices. (b) Performance on networks with 5000 and 10 000 vertices. PreFastMod means the procedure that combines network preprocessing with FastMod, and PreEigenMod represents the procedure that combines network preprocessing with EigenMod.

this set of networks is shown in Fig. 7(a). Since our strategy for calculating accuracy will penalize heavily the results with wrong prediction of the number of communities, we can see from the figure that EigenMod and FastMod cannot correctly detect communities in these networks even if the networks have clear community structure ($0.35 \leq \mu \leq 0.5$). The reason is that resolution limit occurs when optimizing modularity on the partitions of these networks. Conversely, after the networks are preprocessed, EigenMod and FastMod can correctly detect communities as long as $\mu \leq 0.6$. The problem of resolution limit becomes even worse when the number of vertices in the network increases and the community sizes are diverse. We further generated two other sets of networks to confirm the comparison. The vertex number of the first set of networks is 5000 each, while that of the second one is 10 000. The parameters for generating these larger networks are the same as those for 1000-vertex networks except that the minimal community size and the maximal community size in the 10 000-vertex networks are, respectively, 20 and 200 to make the community size more diverse. The results are shown in Fig. 7(b). We only presented the performance

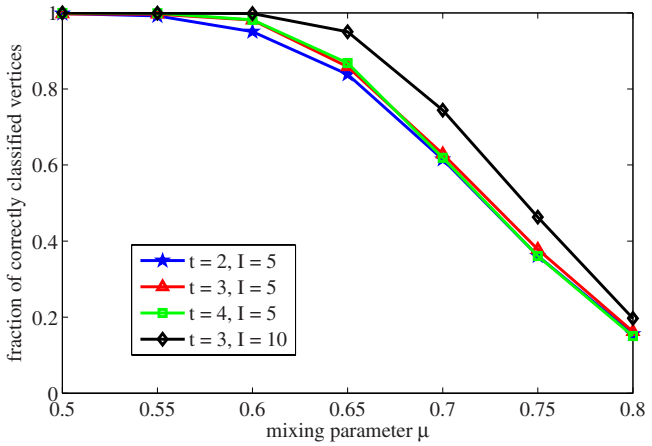

FIG. 8. (Color online) Performance comparison on 5000-vertex LFR networks with different random walk lengths and iterations.

for FastMod, but for EigenMod it is similar. Since the problem of resolution limit becomes more severe in these cases, the performance of FastMod without network preprocessing becomes so bad that it cannot detect communities correctly in the entire range of the mixing parameter. As expected, the performance of FastMod becomes excellent if the networks are preprocessed before they are fed into the modularity-based algorithm. If networks possess clear community structure, different random walk lengths will give very similar results. When $\mu \leq 0.5$, the procedure by network preprocessing with $t=2,3,4$ correctly detects the communities in LFR networks (see Fig. 8). The reason is that in such cases the procedure will quickly set very small (nearly vanishing) weights on inter-community edges (Fig. 7). When $\mu > 0.5$ (i.e., the external degree of a vertex is larger than the internal one), community structures of LFR networks become fuzzier, and it is much harder for the procedure to differentiate intra-community edges from intercommunity ones. As a consequence, weights for real inter-community edges cannot be quickly reduced and more iterations are needed. For example, when we iterate 10 times to weight edges of LFR networks with 5000 vertices, the procedure with $t=3$ can correctly detect communities as long as $\mu \leq 0.6$, showing higher performance than that with 5 iterations (Fig. 8, $\mu > 0.55$).

V. DISCUSSION AND CONCLUSIONS

In this paper, we have proposed a combined procedure to enhance the ability of modularity to detect communities in complex networks. The proposed procedure combines previously developed community detection algorithms with random walk network preprocessing and aims at alleviating the resolution limit problem of modularity optimization. By analyzing the resolution limit of modularity optimization in weighted undirected networks, as the simple extension of the similar analysis in undirected binary ones, we find that the resolution limit of modularity optimization can in theory be overcome as long as we can iteratively reweigh differently intra- and inter-community edges. Network preprocessing is thus used to iteratively put much heavier weights on possible

intracommunity edges but far lighter weights on intercommunity ones. The processed network is then fed into the adopted modularity-based community detection algorithm to produce the partition with the highest modularity, which corresponds to the community decomposition for the original network. We applied the combined procedure to sets of real and synthetic networks. The experimental results demonstrate that the procedure can detect communities as natural as possible and alleviate the resolution limit problem of modularity optimization.

To preprocess the network, we use random walk as a surrogate of dynamic process on it to explore its local structure, which gives us enough knowledge to differentiate intracommunity from intercommunity edges. We use the sum of transition matrices of random walks within path length 3 or 4 to induce a quantity matrix. Each row of this matrix is a vector to be used as the behavior pattern of a vertex. Due to the exponential convergence speed of random walk, small random walk length is empirically chosen to capture enough information on local structure without over-determination and should be not greater than $\log(n)$, with n being the number of vertices in the network. Based on the fact that random walks triggered on vertices in the same community have similar trajectories, an edge is likely to be an intracommunity edge if the behavior patterns associated with the two vertices it connects are very similar, and the likelihood is determined by the cosine similarity between patterns. All the edges in the network are weighted by such likelihoods, and a new weighted network with much clearer community structure than the original one is induced. Iteratively performing edge weighting for a few rounds gives more and more knowledge on intra- and intercommunity edges, this knowledge can then be used by previously developed modularity-based community detection algorithms to find more natural communities. We combined the network preprocessing with several state-of-the-art modularity-based algorithms. Improvements on the results produced by these algorithms indicate the effectiveness of the proposed procedure.

Although modularity presents a resolution limit that prevents it to some extent from being effective in all applications, its effectiveness can be enhanced by combining modularity optimization with a suitable strategy as the random walk network preprocessing suggested here. The benefit coming from the procedure is that we do not need to develop a completely different community detection algorithm but it is possible to adopt any already developed method without modification and with limited computational extra cost. We expect the proposed procedure to be applied to a variety of networks, including social and biological ones.

ACKNOWLEDGMENTS

The authors thank M. E. J Newman for providing Zachary Karate club network and American football network data. The authors also thank M. E. J Newman, A. Arenas, S. Fortunato, R. Lambiotte, and R. Guiméra, for sharing the codes or tools with us. This work is supported by National Natural Science Foundation of China under Grant No. 60873133.

[1] R. Albert and A. L. Barabási, Rev. Mod. Phys. **74**, 47 (2002).
[2] M. E. J. Newman, SIAM Rev. **45**, 167 (2003).
[3] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang, Phys. Rep. **424**, 175 (2006).
[4] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee, IEEE Computer **35**, 66 (2002).
[5] M. Girvan and M. E. J. Newman, Proc. Natl. Acad. Sci. U.S.A. **99**, 7821 (2002).
[6] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Nature (London) **435**, 814 (2005).
[7] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
[8] M. E. J. Newman, Phys. Rev. E **74**, 036104 (2006).
[9] L. Donetti and M. A. Munoz, J. Stat. Mech.: Theory Exp. **2004**, P10012.
[10] S. Fortunato, V. Latora, and M. Marchiori, Phys. Rev. E **70**, 056104 (2004).
[11] R. Guimerà and L. A. Nunes Amaral, Nature (London) **433**, 895 (2005).
[12] J. Duch and A. Arenas, Phys. Rev. E **72**, 027104 (2005).
[13] S. White and P. Smyth, *SIAM Data Mining Conference* (SDM'05, Newport Beach, CA, 2005) p. 274.
[14] M. E. J. Newman, Proc. Natl. Acad. Sci. U.S.A. **103**, 8577 (2006).
[15] S. Fortunato, Phys. Rep. **486**, 75 (2010).
[16] S. Fortunato and M. Barthélemy, Proc. Natl. Acad. Sci. U.S.A. **104**, 36 (2007).
[17] J. Ruan and W. Zhang, Phys. Rev. E **77**, 016104 (2008).
[18] Z. Li, S. Zhang, R. S. Wang, X. S. Zhang, and L. Chen, Phys. Rev. E **77**, 036109 (2008).
[19] J. Reichardt and S. Bornholdt, Phys. Rev. E **74**, 016110 (2006).
[20] J. M. Kumpula, J. Saramäki, K. Kaski, and J. Kertész, Eur. Phys. J. B **56**, 41 (2007).
[21] A. Arenas, A. Fernández, and S. Gómez, New J. Phys. **10**, 053039 (2008).
[22] R. Lambiotte, J. Delvenne, and M. Barahona, e-print arXiv:0812.1770.
[23] T. S. Evans and R. Lambiotte, Phys. Rev. E **80**, 016105 (2009).
[24] J. C. Delvenne, S. N. Yaliraki, and M. Barahona, e-print arXiv:0812.1811.
[25] S. Van Dongen, Ph.D. thesis, University of Utrecht, 2000.
[26] D. Harel and Y. Koren, Lect. Notes Comput. Sci. **2245**, 18 (2001).
[27] M. E. J. Newman, Phys. Rev. E **70**, 056131 (2004).
[28] A. Arenas, J. Duch, A. Fernandez, and S. Gómez, New J. Phys. **9**, 176 (2007).
[29] H. Zhou, Phys. Rev. E **67**, 041908 (2003).
[30] H. Zhou, Phys. Rev. E **67**, 061901 (2003).
[31] H. Zhou and R. Lipowsky, Lect. Notes Comput. Sci. **3038**, 1062 (2004).

[32] P. Pons and M. Latapy, Lect. Notes Comput. Sci. **3733**, 284 (2005).
[33] D. Lai, H. Lu, and C. Nardini, Physica A **389**, 2443 (2010).
[34] W. W. Zachary, J. Anthropol. Res. **33**, 452 (1977).
[35] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre, J. Stat. Mech.: Theory Exp. **2008**, P10008.
[36] J. D. Noh and H. Rieger, Phys. Rev. Lett. **92**, 118701 (2004).
[37] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. (Academic, New York, 1990).
[38] A. Lancichinetti and S. Fortunato, Phys. Rev. E **80**, 016118 (2009).