Identifying network communities with a high resolution

Jianhua Ruan[1,*] and Weixiong Zhang[1,2,†]
[1]*Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, Missouri 63130, USA*
[2]*Department of Genetics, Washington University in St. Louis, St. Louis, Missouri 63130, USA*
(Received 30 April 2007; revised manuscript received 5 October 2007; published 14 January 2008)

Community structure is an important property of complex networks. The automatic discovery of such structure is a fundamental task in many disciplines, including sociology, biology, engineering, and computer science. Recently, several community discovery algorithms have been proposed based on the optimization of a modularity function (Q). However, the problem of modularity optimization is NP-hard and the existing approaches often suffer from a prohibitively long running time or poor quality. Furthermore, it has been recently pointed out that algorithms based on optimizing Q will have a resolution limit; i.e., communities below a certain scale may not be detected. In this research, we first propose an efficient heuristic algorithm QCUT, which combines spectral graph partitioning and local search to optimize Q. Using both synthetic and real networks, we show that QCUT can find higher modularities and is more scalable than the existing algorithms. Furthermore, using QCUT as an essential component, we propose a recursive algorithm HQCUT to solve the resolution limit problem. We show that HQCUT can successfully detect communities at a much finer scale or with a higher accuracy than the existing algorithms. We also discuss two possible reasons that can cause the resolution limit problem and provide a method to distinguish them. Finally, we apply QCUT and HQCUT to study a protein-protein interaction network and show that the combination of the two algorithms can reveal interesting biological results that may be otherwise undetected.

DOI: 10.1103/PhysRevE.77.016104 PACS number(s): 89.75.Hc, 05.10.−a, 02.70.Hm, 87.14.E−

I. INTRODUCTION

Many complex systems can be represented as networks, where vertices are the elements in a system and edges represent relationships between pairs of elements. Examples include social networks [1], genetic networks [2], and the Internet [3]. Much effort has been devoted to the study of topological properties that are common to many networks, such as the small-world property, power-law degree distributions, and high clustering coefficients [4,5].

Another important property of complex networks that has drawn a great deal of attention recently is the so-called community structure—i.e., the existence of some natural division of a network such that the vertices in each subnetwork are highly associated among themselves, while having relatively fewer and weaker connections with the rest of the network [6,7]. Because communities are relatively independent of one another structurally, it is believed that each of them may correspond to some fundamental functional unit. For example, a community in genetic networks often contains genes with similar functions and a community on the World Wide Web may correspond to web pages related to similar topics. Identifying and analyzing such communities from a large network, therefore, provides a means for functional dissection of the network and sheds light on its organizational principles. Furthermore, community structures may provide key insights into some uncharacterized properties of a system. For example, attempts have been made to identify and characterize communities (called functional modules sometimes) in biological networks, leading to *in-silicon* predictions of the functions of some genes [8–10].

Community discovery is similar but not equivalent to the conventional graph partitioning problem [11], both of which require clustering vertices into groups. In the conventional graph partitioning problem, the graph is assumed to be always partitionable and the number of partitions is usually predefined. The challenges in community discovery, however, are twofold: (i) what constitutes a community and (ii) how to effectively find such communities. Although several definitions of communities have been proposed [12,13], none has been universally accepted. The general agreement is that a community discovery algorithm needs to decide by itself the most appropriate community structure without prior knowledge about a network and should be able to distinguish between networks having good community structures and networks with essentially random structures. For an excellent review, see Ref. [14].

Instead of explicitly defining communities, Newman and Girvan recently proposed a quantitative measure, called modularity (Q), to assess the quality of a community structure and formulated community discovery as an optimization problem [15]. The idea has since been widely adopted, and several algorithms have been developed to optimize Q, with good performance in practice [6,16–19]. However, it has been shown that optimizing Q is NP-hard [20], which means an efficient optimal algorithm for the problem is unlikely to exist. The fastest algorithm available uses a greedy strategy and suffers from poor quality [19]. A more accurate method is based on simulated annealing, which has a prohibitively long running time for large networks [18]. The best existing algorithm in terms of both efficiency and effectiveness is due to Newman [6]. A comparison of the performance of some

*Present address: Department of Computer Science, University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249. jruan@cs.utsa.edu
†zhang@cse.wustl.edu

existing methods has been provided in Ref. [21].

On the other hand, although empirical studies have shown that modularity optimization is often an effective way to detect communities in relatively smaller networks, several researchers have observed that this strategy may lead to a resolution limit problem for larger networks [22,23]. Briefly, by optimizing modularity, communities that are smaller than a certain scale or have relatively high intercommunity connectivities may be merged into a single community. This limit, therefore, has cast some doubts on the effectiveness of modularity optimization for community discovery [22].

In this paper, we first present an efficient heuristic algorithm, called QCUT, to optimize Q by combining spectral graph partitioning and local search. We show that the algorithm is able to find higher Q values and is more scalable to large networks than the best existing algorithms. For synthetic networks without the resolution limit problem, we also show that QCUT can achieve a much higher accuracy than the existing algorithms in recovering the known communities.

More importantly, we show that, although modularity optimization has a resolution limit, it is effective in detecting communities at a coarse-grained level; i.e., vertices belonging to the same community tend to be grouped together. This observation is the key for our second algorithm, called HQCUT, to solve the resolution limit problem. The HQCUT algorithm recursively applies QCUT to divide a community into subcommunities. In order to avoid overpartitioning, we use a statistical test to determine whether a community indeed contains intrinsic subcommunities. We demonstrate the effectiveness of HQCUT on a number of synthetic and real networks and show that HQCUT can successfully detect communities at a much finer scale and with a higher accuracy than the algorithms based on modularity optimization alone.

Furthermore, we discuss two primary causes of the resolution limit problem in practice. First, real-world networks often have diverse community sizes. Some small communities may accidentally connect to one another by a few edges due to noises. Second, real-world networks may have hierarchical community structures; i.e., a community may contain several relatively highly interconnected subcommunities. It is crucial to be able to discern such subtle community structures. Therefore, we propose a statistical test to differentiate the two cases and show some interesting statistics in real-world networks.

Finally, we apply QCUT and HQCUT to study a protein-protein interaction network in the budding yeast and analyze the biological significance of the resulting communities. We show that combining the results of these two algorithms can reveal some interesting biological results that may be otherwise undetected.

The rest of the paper is organized as follows. In Sec. II, we introduce an algorithm QCUT for optimizing Q. In Sec. III, we discuss two scenarios that may cause the resolution limit problem for QCUT and other modularity-optimizing algorithms, and propose another algorithm HQCUT to address this issue. We also introduce a method to distinguish the two causes for the problem. We discuss several closely related methods in Sec. IV and present our experimental results in Sec. V. Finally, we conclude with some discussions in Sec. VI.

II. COMMUNITY IDENTIFICATION BY MODULARITY OPTIMIZATION AND THE QCUT ALGORITHM

Let $G=\{V,E\}$ be an undirected network, where V is a set of n vertices and E a set of m edges. The adjacency matrix A of the network is given by

$$A_{uv} = \begin{cases} 1 & \text{if } (u,v) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

We restrict our attention to undirected networks; therefore, A is symmetric. We also assume the network is unweighted, but the definitions below and our algorithms can be easily extended to weighted networks as well. The number of edges, d^u, connected to a vertex u—i.e., the degree of u—can be given by

$$d^u = \sum_{v=1}^{n} A_{uv}. \tag{2}$$

Consider that an algorithm has partitioned the vertices in V into k mutually exclusive groups, $c_1, c_2, \ldots, c_k$. Define

$$e_{ij} = \sum_{u \in c_i, v \in c_j} A_{uv} \tag{3}$$

and

$$a_i = \sum_{j=1}^{k} e_{ij} = \sum_{u \in c_i} d^u, \tag{4}$$

where e_{ij} represents the number of edges connecting the vertices in community i and those in community j, and a_i is the total degree for the vertices in community i. When $i=j$, e_{ii} represents *twice* the total number of edges with both ends in community i. Let $M=2m$ be twice the total number of edges in G. It is easy to see that

$$M = 2m = \sum_{u,v} A_{uv} = \sum_{u} d^u = \sum_{i=1}^{k} a_i = \sum_{i=1}^{k} \sum_{j=1}^{k} e_{ij}. \tag{5}$$

According to Newman and Girvan [15], the modularity of G given a particular partition is defined as

$$Q = \sum_{i=1}^{k} \left[\frac{e_{ii}}{M} - \left(\frac{a_i}{M} \right)^2 \right]. \tag{6}$$

The first term in Eq. (6), e_{ii}/M, measures the fraction of edges falling inside community i, while the second term $(a_i/M)^2$ is the expected fraction of such edges if the edges in the network were randomly rewired with d^u fixed for every u [15]. Therefore, a larger Q value indicates more intracommunity edges that would be expected by chance and hence stronger community structure. For a trivial partition with a single community, $Q=0$. The expected Q value for randomly partitioning a network is also 0. On the other hand, a random network can have a positive or even substantial modularity, especially for sparse networks [24].

Given the definition of Q, the community discovery problem is to find a partition of the network that optimizes Q. As we mentioned in Introduction, however, it has been shown that optimizing Q is NP-hard [20], which means an efficient

optimal algorithm for the problem is unlikely to exist. Here, we propose a heuristic algorithm, named QCUT, to approximately optimize Q. The QCUT algorithm consists of two alternating stages: namely, partitioning and refinement. In the partitioning stage, a spectral graph algorithm is applied to recursively divide a network, until no improvement to Q can be achieved. This step provides an efficient approximate solution to find a reasonably good Q. In the refinement stage, the algorithm attempts to move vertices across communities or merge communities in order to improve Q. The algorithm then returns to the first stage to check if any communities affected by the refinement stage can be further partitioned. These two stages are alternately carried out until neither of them can improve Q.

A. Recursive spectral graph partitioning

The core of the first stage of QCUT is a recursive spectral graph partitioning algorithm, called KCUT, which we have developed earlier [25]. The KCUT algorithm is intermediate between a recursive spectral bipartitioning algorithm [26] that recursively divides each (sub)network into two and a K-way partitioning algorithm that directly divides a network into K subnetworks [17]. The KCUT algorithm is recursive in essence. However, instead of searching for the best bipartitioning at each step, the algorithm attempts to divide a (sub-)network into $2,3,\ldots,l$ subnetworks using a K-way spectral graph partitioning algorithm, where l is a small integer. These partitions are compared, and the one with the highest Q is selected. The algorithm is then applied recursively to each subnetwork, until the Q value of the network[1] cannot be improved.

Given a network G and a small integer l that is the maximal number of partitions to be considered for each step, the algorithm KCUT consists of the following steps.

(1) Initialize C to be a single community with all vertices, and set $Q=0$.

(2) For each community c in C,

(a) let g be the subnetwork of G induced by the vertices in c, and

(b) for each integer k from 2 to l,

(i) find a k-way partition of g, denoted by c_k^g, with a K-way spectral partitioning algorithm, and

(ii) the new community structure, if c_k^g is accepted, is $C_k'=(C\backslash c)\cup c_k^g$, and the new Q value would be $Q_k'=Q(C_k',G)$.

(c) Find the k that gives the highest Q value—i.e., $k^*=\arg\max_k Q_k'$.

(d) If $Q_{k^*}'>Q$, let $C=C_{k^*}'$ and set $Q=Q_{k^*}'$; otherwise, advance to the next cluster in C.

In step (b) above, we utilize an algorithm modified from Ref. [17], which in turn uses a standard K-way spectral clustering algorithm by Ref. [27]. Basically, for a given graph and l, the l largest eigenvectors of a normalized Laplacian matrix of the graph is computed. The eigenvectors are then stacked to form an n by l matrix, where n is the number of vertices in the network and l is the maximum number of partitions to consider at each iteration. Then the first k ($k=2$ to l) columns were taken to partition the rows of the matrix into k groups using a fast k-means algorithm [28].

A more thorough discussion and results of the KCUT algorithm can be found in Ref. [25]. It has been shown that the algorithm has a better accuracy than spectral bipartitioning algorithms and is much faster than the direct K-way partitioning in Ref. [17].

B. Refinement

In the refinement stage, a local search strategy is applied to improve Q as much as possible. We repeatedly consider two types of operations: (i) *migration*: move a vertex from its current community to another one; (ii) *merge*: combine two communities to form a single one. In this process, we use the steepest ascent hill climbing heuristic; i.e., the algorithm always executes the operation that gives rise to the highest Q. Ideally, we can add a split operation which divides a community into smaller ones. However, it is much more expensive to search for a good split than for a migration or merge. Therefore, we consider split only if no migration and merge can improve Q. This is achieved by returning to the partitioning stage.

To efficiently identify a good migration or merge operation, we precompute the change to Q for each potential migration or merge. Let $\{c_1,c_2,\ldots,c_k\}$ be the current best community structure of G and vertex v be a member of community c_i. It can be shown that (see Ref. [29]) the change to Q incurred by moving v to a new community c_j can be computed by

$$\Delta Q^{migr}(v,i,j)=\begin{cases}\dfrac{2}{M}(d_j^v-d_i^v)+\dfrac{2d^v}{M^2}(a_i-a_j-d^v) & \text{if } i\neq j,\\ 0 & \text{if } i=j,\end{cases} \tag{7}$$

where d_i^v is the number of connections that v has in community i and is given by

$$d_i^v=\sum_{u\in c_i}A_{uv}. \tag{8}$$

An intuitive interpretation of Eq. (7) is straightforward: in order to improve Q, we would prefer to move v to a community in which v has more friends (i.e., $d_j^v>d_i^v$) and which is relatively smaller (i.e., $a_j<a_i-d^v$). Given an initial partition, we compute all values of $\Delta Q^{migr}(v,i,j)$ for all pairs of v and j (i is not a free parameter, as it is determined by the current community structure) and store them in a table $T=(t_{vj})_{n\times k}$, where n is the number of vertices, k the number of communities, and t_{vj} the change to Q if v is moved to c_j. T can be efficiently computed with matrix algebra. It may first seem that the table is a dense matrix, taking $O(nk)$ space to store and $O(nk)$ time to search. However, we do not need to store any negative values. Furthermore, it can be shown that we also do not need to compute t_{vj} when $d_j^v=0$, since the corresponding migration will not give the highest ΔQ, even if it is positive (see Ref. [29]). Therefore, for a sparse net-

[1]The Q value here refers to the modularity of the entire network instead of the subnetwork being partitioned in each recurrence.

work, most entries in T can be set to zero, resulting in a sparse matrix.

Similarly, the potential change to Q if c_i and c_j are merged can be computed by

$$\Delta Q^{merge}(i,j) = \begin{cases} \frac{2}{M}\left(e_{ij} - \frac{a_i a_j}{M}\right) & \text{if } i \neq j, \\ 0 & \text{if } i = j, \end{cases} \tag{9}$$

where e_{ij} is the number of edges connecting the two communities. Based on this equation, we can compute a table $S = (s_{ij})_{k\times k}$, where $s_{ij} = \Delta Q^{merge}(i,j)$.

Given S and T, the algorithm then chooses an operation that can result in the highest improvement to Q. This continues as long as there is some positive entry in S or T. Importantly, after an operation is taken, we do not need to recompute the two tables entirely, since most of the entries in S and T remain unchanged. It is easy to see that only the vertices in the refined communities and those connected to the migrated vertices need to be updated. Furthermore, as can be seen from Eqs. (7) and (9), each operation will improve Q by at least $\frac{2}{M^2}$, assuming edge weights are integers. Therefore, the algorithm will terminate in at most $M^2/2$ iterations, while in practice it usually terminates much sooner.

III. LIMITATION OF MODULARITY OPTIMIZATION AND A SOLUTION

Equation (9) implies that any two communities c_i and c_j in a community structure with optimal Q must satisfy

$$e_{ij} \leq \frac{a_i a_j}{M}, \tag{10}$$

where the left-hand side is the number of edges between c_i and c_j and the right-hand side is the expected number of such edges in an equivalent random network. If this condition does not hold, c_i and c_j can be simply merged to improve Q. This condition is intuitively reasonable: when two subnetworks are connected by a higher-than-expected number of edges, they are probably related and therefore should not be separated.

However, consider the network in Fig. 1(a), where two cliques are connected by a single edge. If there are no other vertices, the two cliques clearly form two communities. It becomes interesting, however, when one of the cliques is connected to a large network via a single edge. When the number of edges in the network is above a certain threshold, such that $M > a_i a_j$, the expected number of edges between the two cliques becomes less than 1. Consequently, the two cliques will be considered as a single community, according to Eq. (9). In other words, by optimizing modularity, communities smaller than a certain scale tend to be merged with other communities. This phenomenon has received attention recently and been referred to as the resolution limit problem in Ref. [22].

This resolution limit has some significant impact in practice. First, real-world networks often contain both large and small communities. In addition, many real-world networks such as social or biological networks are constructed from

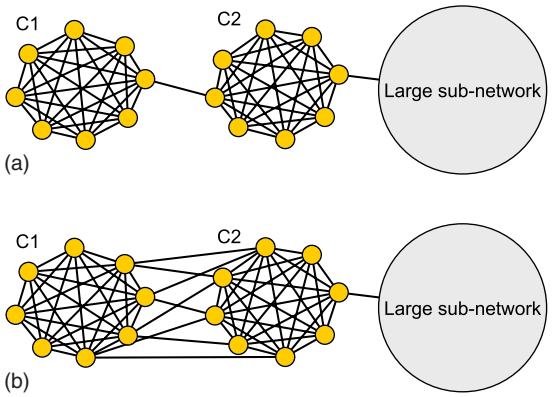

FIG. 1. (Color online) Networks with possible resolution limit problems. (a) Two loosely linked cliques connected to a much larger subnetwork via a single edge. (b) Two relatively densely linked cliques connected to a much larger subnetwork via a single edge. In either case, when the subnetwork on the right-hand side is sufficiently large, the two cliques will be identified as a single community due to the resolution limit. However, these two cases may have different meanings. In (a), the two cliques may represent two independent communities linked together by chance or errors. In (b), the two cliques may represent two subcommunities of a larger community.

survey or experimental data and therefore often contain errors. If two small communities are accidentally connected by a false edge, they will be nonseparable by modularity optimization. The limitation, therefore, is partially due to the assumption that all edges in a network are reliable. Furthermore, the modularity function is also limited by the implicit assumptions that the entire community structure of a network has no hierarchy and that a vertex can freely connect to any other vertex in the network. Consider the network in Fig. 1(b). The two cliques are connected by a relatively large number of edges, which are unlikely due to chance. Therefore, the two cliques can be considered as a single community. On the other hand, it is evident that the edge density between the two cliques is much smaller than that within the cliques, indicating substructures within the community. In reality, the concept of communities may vary, depending on at what granularity the network is analyzed. For example, from the viewpoint of the General Secretary of the United Nations, each country may be a community, while from the viewpoint of an elementary school student, her definition of community may correspond to the classes in her school.

It is important to note the intrinsic difference between the scenarios in Figs. 1(a) and 1(b). In Fig. 1(a), the two subnetworks cannot be separated due to their small sizes relative to the entire network. Although the number of edges connecting the two components is higher than expected, the difference between the observed and expected number of intercommunity edges is not statistically significant; i.e., the intercommunity edges may have appeared just by chance. Therefore, we call the two subnetworks *affiliated*. On the other hand, in Fig. 1(b), the two subnetworks are statistically closely associated, which may indicate some functional relationships. Therefore, we call them *associated*. Note that, however, there is no clear distinction between the two types of intercommunity relationships.

A. HQCUT algorithm

In order to address the resolution limit problem of the modularity function, Fortunato and Barthelemy suggested a method that applies modularity optimization to each subnetwork to identify subcommunity structures [22]. Here we generalize the idea. We first apply QCUT to obtain a community structure with the highest Q. We then apply QCUT to each subnetwork *recursively*, while ignoring all the intercommunity edges. A critical issue is, then, how to decide whether a community should be further partitioned or not.

Here we propose two criteria. First, if the modularity[2] of partitioning a subnetwork is below a threshold min q, it is an indication that the subnetwork has no strong subcommunity structure and therefore should not be partitioned. Second, it has been shown that a network may have a high modularity by chance, especially if the network is sparse [24]. To address this problem, we estimate the statistical significance of the modularity using a Monte Carlo method. For each subnetwork, we apply QCUT to obtain a maximal modularity q. The subnetwork is then randomly rewired with a procedure described in Ref. [30] to obtain N random subnetworks, where each vertex has the same degree as in the original subnetwork. The QCUT algorithm is applied to each random subnetwork to search for a maximal modularity. We compute the statistical significance of q using a Z score:

$$Z = \frac{q - \langle q \rangle}{\sigma_q}, \tag{11}$$

where $\langle q \rangle$ and σ_q are the mean and standard deviations of the modularity values of the random subnetworks. A high Z score indicates a statistically significant modularity of the subnetwork and therefore may correspond to real subcommunity structures.

Most real-world networks have $Q \geq 0.3$ [5]. Therefore, we use this value as the default value of min q. This cutoff prevents statistically significant but practically uninteresting partitions to be considered (for example, a high Z score could be achieved if σ_q is very small). Second, we use a Z score cutoff min $Z \geq 2$, which corresponds to a p value of 0.05. As shown in Ref. [29], the results are generally insensitive with respect to a wide range of parameter values.

B. Differentiate affiliated and associated communities

As we have discussed, both affiliated and associated subcommunities are nonseparable by simply optimizing modularity. HQCUT can identify both types of subcommunities, but is unable to differentiate them. Therefore, after obtaining the result of HQCUT, we need to determine whether two communities are associated or affiliated. For this purpose, we first identify pairs of communities that are directed connected by some edges. Then for each candidate community pair (c_i, c_j) connected by e_{ij} edges, we use both analytical and simulation methods to estimate the probability that we would see at least e_{ij} edges between them if the network is randomly rewired. With the community structure of the network and the degree of each vertex fixed, we randomly rewire the network with the procedure described in Ref. [30] and count the number of edges, $\hat{e}_{ij}$, between communities i and j. The number of times that $\hat{e}_{ij} \geq e_{ij}$ divided by the total number of randomizations is used as the empirical p value. Furthermore, the theoretical p value can also be estimated analytically with a hypergeometric distribution with parameters M, a_i, and a_j. This relationship can be best explained by considering how the network randomization works. Conceptually, to randomly rewire a network, we break each edge into two halves, or stubs, and then randomly reconnect these stubs into edges. Therefore, for a given network with m edges, we have a box of $M=2m$ edge stubs. The probability to observe exactly k edges between community c_i and c_j is equivalent to the probability of randomly drawing a_i stubs from the box and observing k of them connected to the vertices in c_j. This probability is given by

$$f(k;M,a_i,a_j) = \frac{\binom{a_j}{k}\binom{M-a_j}{a_i-k}}{\binom{M}{a_i}}. \tag{12}$$

The p value for observing at least e_{ij} edges between c_i and c_j, therefore, can be computed by

$$P(e_{ij},M,a_i,a_j) = \sum_{k=e_{ij}}^{\min(a_i,a_j)} f(k;M,a_i,a_j). \tag{13}$$

To be precise, Eq. (13) will slightly overestimate the p value due to the constraint that in the randomly rewired networks self edges are prohibited. Our experiments have shown, however, that the p values obtained empirically and analytically are nearly identical. Therefore, the results presented in this work are based on the analytical method.

Given the p value of observing some number of edges between a pair of communities, we compute an *association score* between the two communities by

$$S(c_i,c_j) = -\log_{10} P(e_{ij},M,a_i,a_j). \tag{14}$$

Note that Eq. (14) is also defined for $i=j$, which can be used to measure the statistical significance of a community. We define two communities as *associated* if their association score is greater than 2 (i.e., $p<0.01$) and *affiliated* if their association score is less than 1 (i.e., $p>0.1$). Community pairs with intermediate association scores remain undefined, since we do not have enough statistical evidence about their relationships. Furthermore, we define a community as an *associated community* if it is associated with another community and an *affiliated community* if it is affiliated with another community.

IV. RELATED WORKS

Since the invention of the modularity function, a number of community-finding methods have been developed, based on optimizing Q. Here we briefly discuss several methods

[2]Here the modularity refers to the Q value by treating the subnetwork as an independent network, rather than the Q value of the whole network.

that share some common characteristics with our QCUT algorithm.

Newman recently proposed a method that is also based on spectral graph partitioning and local search [6]. QCUT significantly differs from Newman's algorithm in two aspects. First, for the spectral partitioning, our algorithm utilizes the Laplacian matrix of a network, while his method deals with a so-called modularity matrix. It has been shown that both spectral partitioning methods can approximately optimize Q [6,17]. Since the Laplacian matrix is typically sparse, while the modularity matrix is almost a complete matrix, our algorithm has a much lower memory requirement and is more scalable to large networks. Second, Newman's algorithm uses a Kernighan-Lin heuristic after each partitioning to switch members in two sibling communities. Therefore, the refinement decision in his algorithm is only made locally. In contrast, a vertex can be moved to any communities in the refining stage of our algorithm; therefore, the decision is made globally. Furthermore, QCUT carries out partitioning and refinement alternately, while Newman's algorithm terminates when the partitioning is completed.

Another method proposed by Duch and Arenas is based on a technique called the extremal optimization [31]. In their method, a (sub)network is first randomly partitioned into two and vertices are then moved across communities to improve Q. Similar to our method, it uses a fitness function to guide which vertex should be moved first. However, it only considers moving vertices between two sibling communities and no global structure adjustment is attempted. The simulated annealing algorithm by Ref. [18], on the other hand, allows moving a vertex to any community as in our method and significantly improves the optimization accuracy. However, its stochastic nature makes it infeasible for large networks.

On the other end of the spectrum is a greedy method called the CNM algorithm [19], which takes a completely different direction. Initially, each vertex is treated as a community. Larger communities are formed by merging smaller ones, if such a merge would increase the Q value of the network. Conceptually, the CNM algorithm is very similar to the agglomerative hierarchical clustering in the statistics field [32]. Despite its relatively low accuracy, CNM is very efficient and can be applied to networks with millions of nodes. For very large networks, a practical strategy may be to obtain a fast approximate solution with CNM and use QCUT to further optimize the partitions.

Finally, several studies have been performed to analyze the effect of more realistic characteristics of communities on community identification algorithms. Such characteristics include heterogeneous community sizes [33], hierarchical structures [34], overlapping communities [35], and fuzzy communities [36]. These studies have generally indicated the insufficiency of modularity optimization alone in identifying more complex community structures. A thorough discussion of these results is out of the scope of this paper. Interested readers can follow the cited literature above.

V. RESULTS

In order to test the performance of our algorithms, we applied them to a variety of synthetic or real-world networks and compared them with Newman's algorithm (NEWMAN) [6] as well as the simulated annealing algorithm (SA) [18]. Implementations of NEWMAN and SA were obtained from the original authors.

A. Computer-generated networks

We first considered three sets of computer-generated networks with known community structures and compared the accuracy of the algorithms in identifying the known communities. Each network in these tests has 1000 vertices.

The first set of networks was constructed as follows. The vertices in each network were divided into 20 communities, each with 50 vertices. Edges were randomly placed between the vertices in the same community with a probability p_{in} and across communities with a probability p_{out}. We chose $p_{in}=0.3$, which corresponded to approximately 15 intracommunity edges for each vertex on average ($\langle n_{in}\rangle$), and varied p_{out} such that the average intercommunity edges per vertex, $\langle n_{out}\rangle$, were from 5 to 50. Note that we did not constrain $\langle n_{out}\rangle$ to be smaller than $\langle n_{in}\rangle$ as in many other studies, since we believe what makes a community is its relatively density, not the absolute number of connections. Overall, the networks we used are considerably larger and may include much weaker communities than some of the networks used in other studies.

The second set of networks differed from the first set in that the communities had different sizes. To be precise, each network contained 53 communities, 1 with 100 vertices, 3 with 40 vertices, 9 with 20 vertices, and 40 with 15 vertices. Separate p_{in} and p_{out} were chosen for each community such that each vertex has on average $\langle n_{in}\rangle=6+\ln L$ intracommunity edges, where L is the size of the community that the vertex resides, and $\langle n_{out}\rangle=2$ to 24 intercommunity edges on average. This choice of $\langle n_{in}\rangle$ avoids the problem of having either a fixed p_{in} or a fixed $\langle n_{in}\rangle$ for all nodes. For a fixed p_{in}, the smaller communities may be too fuzzy to be detected or considered significant, while for a fixed $\langle n_{in}\rangle$, the smaller communities may become much denser than the larger communities and easier to identify.

We designed a third set of networks to contain hierarchical communities, following the model proposed in Ref. [34]. The vertices in each network were first grouped into ten equal-sized communities (level-1 communities). Each community was then divided into two subcommunities (level-2 communities). Edges were placed randomly with probability $p_{out}=0.01$ between vertices in different level-1 communities, $p_{in}^1=0.3$ between vertices within the same level-2 community, and $p_{in}^2=0.05$ between vertices within the same level-1 community but in different level-2 communities.

To measure the accuracy of a predicted community structure, we computed the Jaccard index, which is based on the number of correctly identified intracommunity vertex pairs [37]. Given a true community structure C_1 and a predicted community structure C_2, let S_1 be the set of vertex pairs in the same community in C_1 and S_2 the set of vertex pairs in the same community in C_2. The Jaccard index is defined by

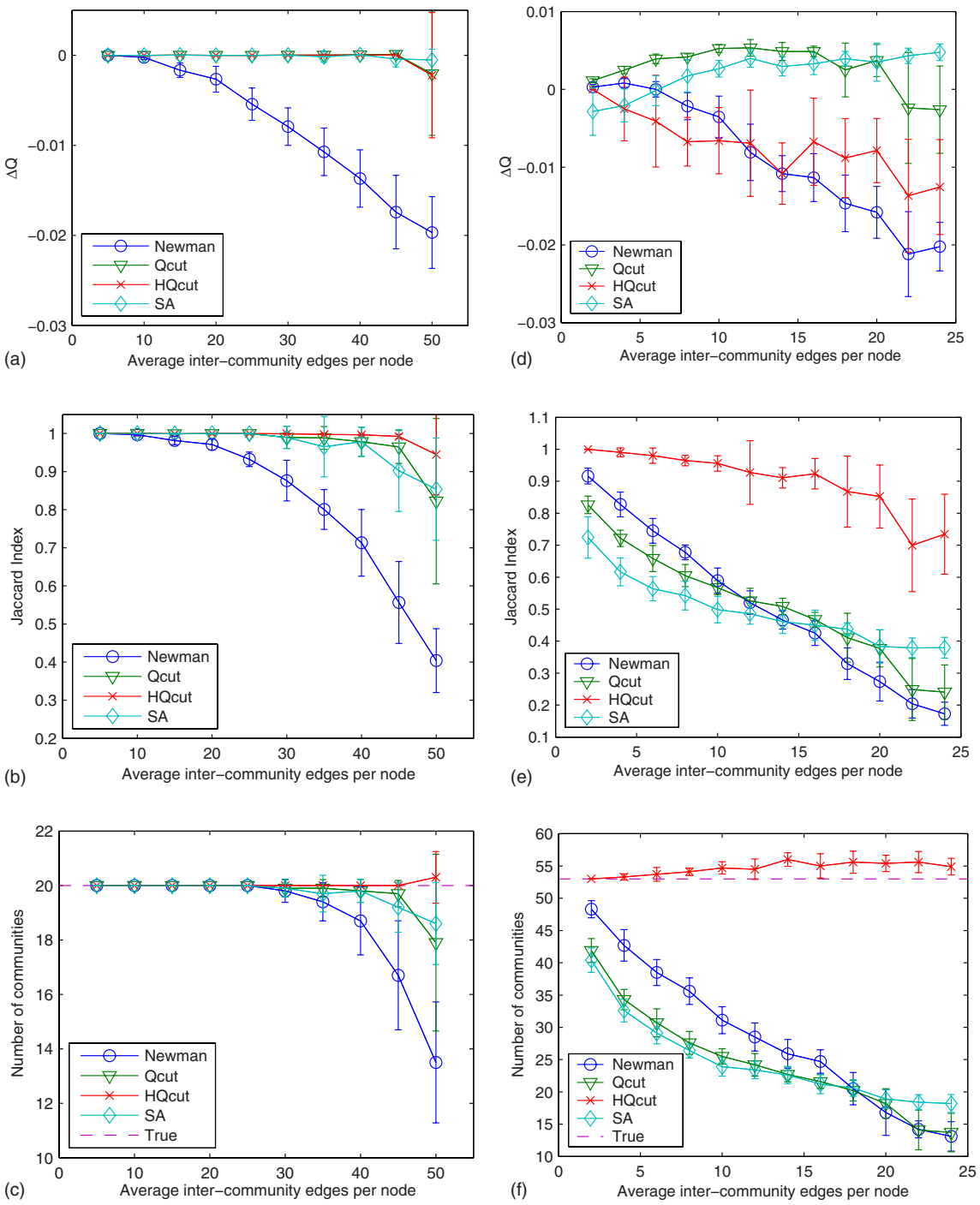

FIG. 2. (Color online) Results on computer-generated networks. ΔQ is the difference between the observed Q value and the expected Q value. (a)–(c) Networks with equal community sizes. (d)–(f) Networks with heterogeneous community sizes. Error bars show the standard deviations estimated from 100 networks.

$$J(S_1,S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}. \tag{15}$$

The value of Jaccard index is within [0, 1], with 1 being the most accurate. The results using two other accuracy measurements, the Fowlkes-Mallows index [38] and variation of information [39], are provided in Ref. [29]. For larger networks, measuring accuracies based on the number of vertex pairs are much more convenient and meaningful than measuring the fraction of correctly identified vertices, as suggested by Ref. [33].

As evident from Figs. 2(a)–2(c), for the first set of networks, QCUT and SA clearly outperformed NEWMAN in optimizing Q. Furthermore, the slightly improved Q values resulted in significantly better accuracies of community structures. NEWMAN often returned fewer communities than it should when the community structures became weaker,

while the other three algorithms performed very well in almost all cases. In addition, when $\langle n_{out} \rangle \leq 35$, QCUT and HQCUT had almost the same results, indicating that HQCUT did not overpartition the communities. For large $\langle n_{out} \rangle$ values, however, HQCUT had slightly lower modularity but higher accuracy than QCUT. Indeed, for these networks, because of the abundance of intercommunity edges, some communities were merged by QCUT due to the resolution limit.

Figures 2(d)–2(f) show the results for the second set of networks. HQCUT was run with the default parameters, and the results were robust with respect to a wide range of parameter values (see Ref. [29]). As shown, QCUT and SA again found better modularities than NEWMAN [Fig. 2(d)]. However, it is clear that for these networks, the higher modularities did not result in better community accuracies. In fact, the modularities found by QCUT or SA were often higher than those of the true community structures [Fig. 2(d)]. HQCUT, on the other hand, has achieved the highest accuracy for all the networks, despite slightly lower modularities [Fig. 2(e)]. HQCUT also recovered almost all the communities, with very few false-positive partitions, for all cases [Fig. 2(f)]. In contrast, the other three algorithms often merged many small communities and returned only half or even one-third of the communities.

For small values of $\langle n_{out} \rangle$, NEWMAN reached slightly better accuracies than QCUT and SA [Fig. 2(e)]. However, the low accuracy of QCUT was primarily caused by the merge of small communities, which can be easily resolved by a recursive algorithm such as HQCUT. In other words, by optimizing Q, we have a better chance to group together pairs of vertices that belong to the same community. In contrast, NEWMAN not only merged some small communities, but also assigned many vertices to wrong communities, which cannot be resolved easily (see Ref. [29]).

As shown in Figs. 2(c) and 2(f), HQCUT may occasionally overpartition some communities, but seldom underpartitions any community. This is true even if we increase min Z to a much larger value (e.g., 10). There may be two reasons for this to occur. First, although the test networks were randomly generated, some communities may have "nonrandomness" simply by chance. In other words, the generated networks may in fact have more communities than what we expected. In this case the results do not represent a failure of the algorithm. The second reason is that, when QCUT was recursively applied to find lower-level communities, the edges between the higher-level communities were completely ignored. Therefore, the algorithm has more freedom to find some local-community-like structures. Those communities typically have many intercommunity edges and can often be detected by measuring the intercommunity association scores and self-association scores with Eq. (14), as shown above. It may be an interesting future direction to develop new algorithms or objective functions that explicitly take into account those intercommunity edges.

For the third set of networks, QCUT and NEWMAN successfully identified all level-1 communities with 100% accuracy, but could not separate the level-2 communities. In comparison, HQCUT further partitioned the network and successfully detected all level-2 communities with an accuracy of 99.9%.

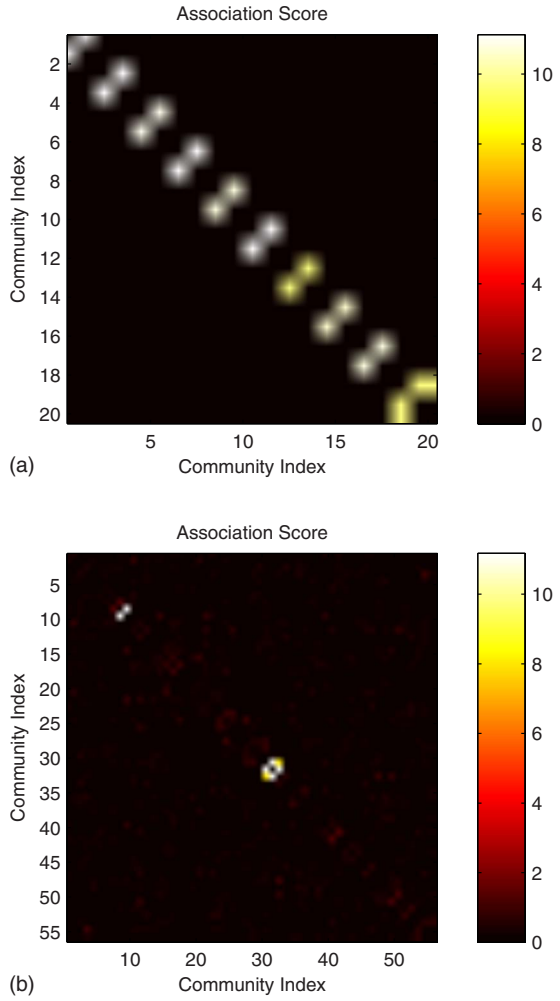

FIG. 3. (Color online) Association scores between pairs of communities identified by HQCUT from (a) a network with hierarchical communities and (b) a network with heterogeneous sizes of communities. Self-association scores, which are always high here, are not shown in order to emphasize on the intercommunity relationships.

Finally, we applied the statistical test procedure described in Sec. III B to evaluate whether it is possible to distinguish the intercommunity relationships in the second set of networks and the third set of networks. Figure 3 shows the association scores for the communities in two networks. The first network is chosen from the third set of networks, while the second network is chosen from the second set of networks, with $\langle n_{in} \rangle = 10$. The communities in the first network are organized into two levels; therefore, significant association scores are expected between pairs of level-2 communities in the same level-1 community. Indeed, as shown in Fig. 3(a), each identified community is strongly associated with a neighboring community, indicating hierarchical community structures in the network. In contrast, the communities in the second network do not have hierarchies, and therefore we would not expect any significant association among them. The HQCUT algorithm identifies 56 communities for this network while the true structure contains 53 communities. The association scores between pairs of communities are shown

in Fig. 3(b). As shown, most communities are not significantly associated with any other communities. There are two groups of communities—i.e., communities 9 and 10, as well as communities 31, 32, and 33—that are significantly associated. The reason is that HQCUT has slightly overpartitioned some communities. If we had combined communities 9 and 10, as well as communities 31, 32, and 33, we would have predicted the true community structure perfectly.

B. Real-world networks

As a further test of our algorithms, we applied them to several real-world networks, which may have different topological properties than the computer-generated networks.

In the first real-world network, each vertex is a football team in the United States NCAA division I-A and an edge between two teams represents a regular-season game played by them in year 2006. This network is interesting because of its known community structure. The 115 teams have been organized into 11 conferences (excluding the teams in the independence conference), and games were played more frequently between teams in the same conference than teams in different conferences. Therefore, each conference can be considered as a community.

Applying QCUT to the network, we discovered eight communities (Q=0.608), five of which matched individual conferences precisely (Pacific-10, Conference USA, Big 12, Sun Belt, and Southeastern) (Fig. 4). Each of the other three communities contains two conferences: one community contains Western Athletic and Mountain West, one contains Big Ten and Mid-American, and the other contains Big East and Atlantic Coast. The teams in these conferences have a relatively high frequency of interconference games with the teams in a conference that are geographically close. NEWMAN returned the same results as QCUT. In contrast, with HQCUT, the network was divided into 11 communities (Q=0.596), each of which corresponds to a conference precisely (Fig. 4).

We also tested the algorithms on a number of real-world networks with unknown community structures. For these tests, we were unable to measure the accuracy of the algorithms due to the lack of known community structures. Therefore, we focused on the modularity values. As we have shown on the synthetic networks, although a higher modularity may not necessarily guarantee a better accuracy in community discovery, it nevertheless generally means better accuracy in recovering the true intracommunity vertex pairs, which is necessary for a recursive algorithm such as HQCUT to succeed.

The results on these networks are shown in Table I. The detailed information of the networks is included in Ref. [29]. As shown, QCUT always obtained higher modularities than NEWMAN. While SA can achieve higher modularity for small networks, its performance on large networks is often worse than QCUT and NEWMAN, even with much longer running time. The NEWMAN algorithm is faster than QCUT on networks up to ~1500 vertices, but slower than QCUT for larger networks.

Next, we applied HQCUT to these networks and compared the results to those in Ref. [22] (SA-2), which were obtained by applying SA to each community while ignoring the intercommunity edges. Although SA-2 only allowed one level of hierarchy while HQCUT supported multiple levels of hierarchy, the latter usually returned fewer subcommunities than the former, indicating that SA-2 had probably overpartitioned these networks.

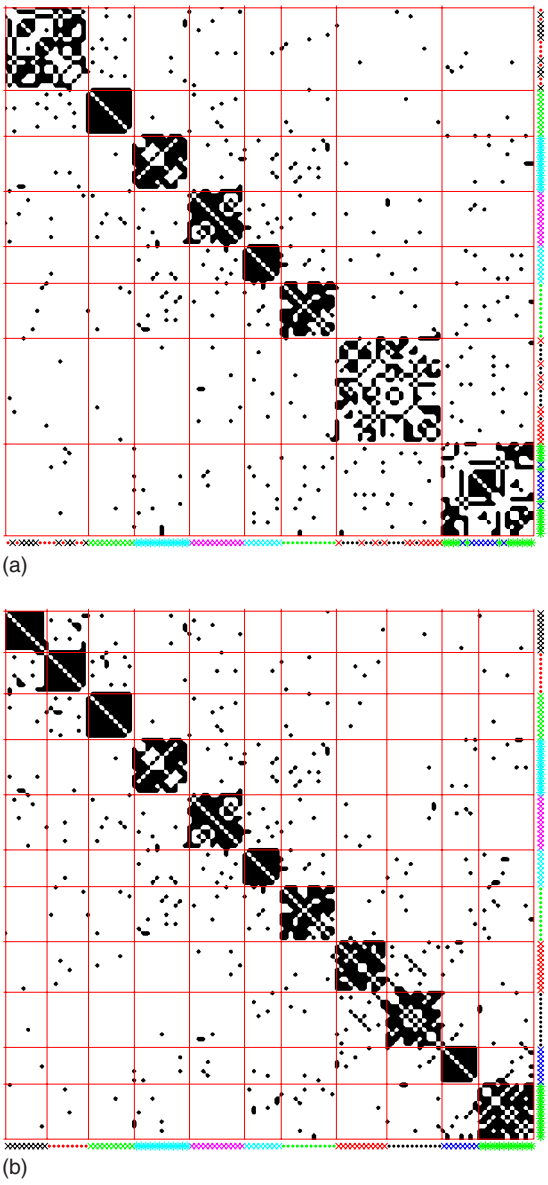

FIG. 4. (Color online) Community structure in the football team network. Each symbol along the axes represents a conference. (a) Results of QCUT. (b) Results of HQCUT. The 11 conferences shown in (b), from top to bottom, are Western Athletic, Mountain West, Pacific-10, Conference USA, Big 12, Sun Belt, Southeastern, Big Ten, Mid-American, Big East, and Atlantic Coast.

In order to test what type of communities are more abundant in the networks, we counted for each network the number of associated or affiliated communities as defined in Sec. III B. Interestingly, as shown in Table I, some networks consist of primarily affiliated communities while other networks contain many associated communities, indicating hierarchical community structures in the latter group of networks.

TABLE I. Community results on real-world networks. n, the number of vertices; m, the number of edges; and k the number of communities. The results by SA and SA-2 on the first five networks were directly obtained from Ref. [22]. The unit of time is seconds, unless "m" (minute) is specified. Time comparison is for illustration only, since the programs were implemented and executed within different computing environment. NEWMAN and SA were implemented in the C programming language and compiled into LINUX binary codes before execution. QCUT was implemented in MATLAB script and compiled at run-time. The last column shows the numbers of affiliated (Af) versus associated (As) communities.

Network			NEWMAN			SA			QCUT			SA-2			HQCUT		
Name	n	m	k	Q	Time	k	Q	Time	k	Q	Time	k	Q	k	Q	Af/As
Social	67	142	8	0.573	0.01	10	0.608	5.4	8	0.587	2	21	0.532	9	0.578	9/0
Neuron	297	2359	4	0.396	0.4	4	0.408	139	4	0.398	1.9	20	0.319	10	0.363	2/6
Ecoli Reg	418	519	38	0.766	0.7	27	0.752	147	39	0.776	12.7	76	0.661	44	0.769	40/0
Circuit	512	819	15	0.804	1.8	11	0.670	143	13	0.815	6.1	70	0.64	43	0.723	9/15
Yeast Reg	688	1079	26	0.759	3	9	0.740	22.5 m	27	0.766	13.4	57	0.677	66	0.696	28/13
Ecoli Met	563	709	29	0.827	2.06	19	0.828	200.4	21	0.835	12	92	0.728	37	0.81	21/2
Ecoli PPI	1440	5871	18	0.367	33.2	14	0.387	97.8 m	21	0.387	41.5	88	0.305	112	0.346	10/31
Internet	3015	5156	46	0.611	253.7	20	0.624	184 m	21	0.634	43	219	0.556	186	0.566	32/59

This preference seems to be unrelated to the edge density or the modularity of the networks and may deserve further studies. A possible explanation is that the edges in the latter group of networks (e.g., circuit, PPI, and Internet) represent physical interactions. As a result, the interactions are limited by some spatial or structural constraints, and therefore a hierarchical community structure may be more feasible. In contrast, the edges in the former group represent some logical relationships and therefore are not limited by such constraints.

C. Application to a biological network

Finally, as a real application, we applied NEWMAN, QCUT, and HQCUT to a protein-protein interaction network and studied the discovered communities in more detail. The network contains 2708 proteins and 7123 pairwise physical interactions in the yeast *Saccharomyces cerevisiae* [40]. NEWMAN and QCUT identified 56 ($Q=0.694$) and 93 communities ($Q=0.696$), respectively, while HQCUT detected 316 communities ($Q=0.582$). In order to determine the biological significance of the communities, we compared the communities to the known protein complexes in the MIPS database [41]. Among the 2708 proteins in the network, 817 appeared in at least one protein complex in the database. The protein complexes in the MIPS database are also organized into some hierarchical structures; i.e., a large protein complex may contain several smaller complexes. A protein may also belong to multiple protein complexes. In order to measure how well a discovered protein community represents real protein complex, we computed a *matching score* for a community c as follows:

$$M(c) = \max_i \frac{|c \cap p_i|}{\sqrt{|c \cap P| \times |p_i \cap C|}}, \tag{16}$$

where p_i is the ith known protein complex and $c \cap p_i$ is the set of proteins shared between c and p_i. C and P represent the set of all proteins in the network and in the MIPS protein complex database, respectively. The overall performance of the algorithm was measured by the weighted average of matching scores for all communities. To eliminate trivial matches between very small communities and small protein complexes, we made the restriction that a match must include at least three proteins. This filter particularly affected HQCUT, which predicted a large number of small communities. Those communities were removed from our analysis to avoid biases, since small communities tend to have high matching scores by chance.

Table II shows some statistics of the matching scores, and Fig. 5 shows the total number of proteins as a function of matching score cutoffs. Overall, 81 communities (553 proteins) identified by HQCUT matched to some known complexes, with a weighted average matching score of 0.81. In comparison, 31 communities (778 proteins) by QCUT and 40 communities (779 proteins) by NEWMAN matched to some known complexes, with average matching scores of 0.55 and 0.58, respectively. Furthermore, HQCUT discovered 28 communities (126 proteins) that perfectly matched to some known protein complexes. In contrast, QCUT and NEWMAN identified much fewer perfect matches between communities and protein complexes. Therefore, by allowing subcommunities, HQCUT has recovered a larger number of real protein complexes, while the communities identified by NEWMAN or QCUT may contain multiple protein complexes.

TABLE II. Statistics of protein community–complex-matching scores.

	NEWMAN	QCUT	HQCUT
No. of communities	56	93	216
No. of matched communities	40	31	81
No. of matched proteins	779	778	553
Weighted average matching score	0.58	0.55	0.81
No. of perfectly matched communities	3	2	28
No. of perfectly matched proteins	16	6	126

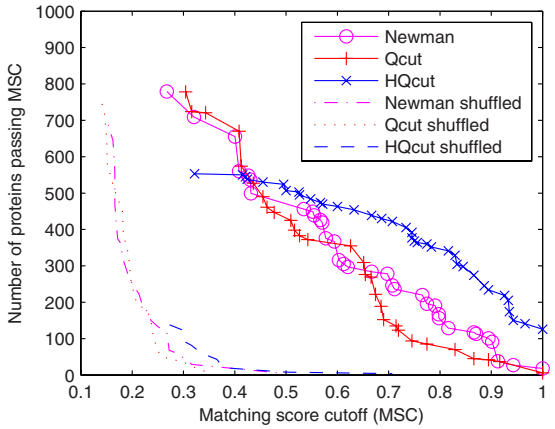

FIG. 5. (Color online) Number of proteins as a function of matching score cutoffs (MSCs).

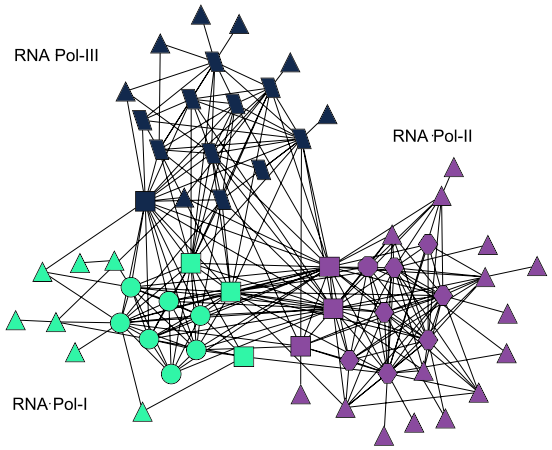

FIG. 6. (Color online) Example hierarchical communities in PPI network. The three colors represent three subcommunities discovered by HQCUT in a community identified by QCUT. Circles, hexagons, and parallelograms are known components of RNA polymerase I, II, and III, respectively. Squares are shared components of two or three RNA polymerases. Triangles are proteins that are not components of the three complexes by current knowledge.

To further estimate the bias introduced by community size differences, we created random community structures that have the same community size distributions as the real ones by randomly shuffling members among communities. The same analysis above was repeated on those shuffled random communities. Indeed, if we had not filtered out matches with less than three overlapping proteins, the random counterpart of the HQCUT communities can have a significant number of high-scoring communities (approximately 50 proteins with matching scores >0.5 and 30 proteins with matching scores >0.6, data not shown). As mentioned, this is expected since HQCUT produces many small communities which may have high matching scores with a small protein complex simply by chance. After applying the filter, however, the random communities for the three algorithms have similar matching scores and none of them are likely to have matching scores above 0.5, indicating that our comparison are not affected by the sizes of communities (Fig. 5).

We again computed the numbers of affiliated and associated communities in this network and found that it contains more associated communities than affiliated ones (195 vs 83), indicating that the majority of the additional communities found by HQCUT are due to hierarchical communities. To analyze whether the hierarchical structures have any biological significance, we manually inspected the associated communities. Interestingly, almost all of the statistically significantly associated communities are biologically related. For example, the three ribonucleic acid (RNA) polymerases, RNA pol I, RNA pol II, and RNA pol III were identified as a single community by QCUT, but were further partitioned by HQCUT into three subcommunities (Fig. 6). The three communities are highly associated due to a few common components shared by the three complexes. In another example, snRNA subunits U1, U2, and U6 were also identified as a single community by QCUT but separated into three subcommunities by HQCUT. Other examples include SAGA and TFIID complexes, INO80 and SWR1 complexes, as well as eIF-2B and eIF-3 complexes (see Ref. [29]). Therefore, by combining the results of QCUT and HQCUT, we are able to reveal the true hierarchical community structures of the network.

VI. CONCLUSIONS AND DISCUSSION

In this paper, we described an efficient algorithm QCUT for discovering communities from complex networks by optimizing the modularity function. We showed that the algorithm can find higher modularities than the existing algorithms on both computer-generated and real-world networks. When the communities are not so small and the intercommunity connectivities are sparse, a higher modularity indeed represents a better community discovery accuracy. On the other hand, we also showed that, when a network contains small or hierarchical communities, optimizing modularity may fail to reveal the fine structures at a satisfactory resolution. To circumvent this problem, we proposed a recursive algorithm HQCUT, which provides a higher resolution without introducing spurious communities. Using a variety of synthetic as well as real-world networks with known community structures, we demonstrated that HQCUT can achieve a much higher accuracy than algorithms based on modularity optimization alone. We also studied a protein-protein interaction network, and found that the protein communities identified by HQCUT correspond to known protein complexes very well, while each community found by modularity optimization may contain several protein complexes.

Our results may first seem to suggest that modularity optimization is not a very good strategy for community discovery in practice. Nevertheless, the success of HQCUT largely depends on the effectiveness of QCUT to optimize Q. By optimizing modularity, the QCUT algorithm may merge several communities into a single one, which can be easily separated by a recursive algorithm such as HQCUT in this paper. In contrast, algorithms that did not succeed in optimizing modularity may split the members of a community into several communities, a mistake that cannot be easily recovered in post-processing.

Finally, we proposed a statistical significance test to differentiate the two scenarios that may cause the resolution limit: small communities and hierarchical communities. By combining HQCUT with the significance test, we were able to not only detect communities with a high resolution, but also identify pairs of highly associated communities. As shown in the case of protein-protein interaction networks, these statistically associated community pairs are indeed functionally related and form a community at a higher hierarchy. Since many real-world networks are hierarchical, identifying and analyzing such structures will be an essential step toward understanding their organizing principles in general.

ACKNOWLEDGMENTS

This research was supported in part by NSF Grants No. ITR/EIA-0113618 and No. IIS-0535257 and a grant from Monsanto Co. to W.Z. J.R. was supported by a new faculty startup fund from the University of Texas at San Antonio while revising this manuscript. The authors wish to thank Mark E. J. Newman, Roger Guimera, Luis A. N. Amaral, and Uri Alon for sharing their programs and network data and two anonymous reviewers for their very insightful comments.

[1] M. E. J. Newman, Proc. Natl. Acad. Sci. U.S.A. **101**, 5200 (2004).
[2] G. D. Bader and C. W. Hogue, Nat. Biotechnol. **20**, 991 (2002).
[3] J. Kleinberg and S. Lawrence, Science **294**, 1849 (2001).
[4] R. Albert and A. L. Barabasi, Rev. Mod. Phys. **74**, 47 (2002).
[5] M. E. J. Newman, SIAM Rev. **45**, 167 (2003).
[6] M. E. J. Newman, Proc. Natl. Acad. Sci. U.S.A. **103**, 8577 (2006).
[7] M. E. J. Newman, Phys. Rev. E **74**, 036104 (2006).
[8] J. B. Pereira-Leal, A. J. Enright, and C. A. Ouzounis, Proteins **54**, 49 (2004).
[9] V. Spirin and L. A. Mirny, Proc. Natl. Acad. Sci. U.S.A. **100**, 12123 (2003).
[10] D. M. Wilkinson and B. A. Huberman, Proc. Natl. Acad. Sci. U.S.A. **101**, 5241 (2004).
[11] P. O. Fjällström, Linköping Electronic Articles in Computer and Information Science **3**, 10 (1998).
[12] G. W. Flake, S. Lawrence, C. L. Giles, and F. Coetzee, IEEE Trans. Comput. **35**, 66 (2002).
[13] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, Proc. Natl. Acad. Sci. U.S.A. **101**, 2658 (2004).
[14] M. E. J. Newman, Eur. Phys. J. B **38**, 321 (2004).
[15] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
[16] J. Ruan and W. Zhang, *Proceedings of the 21st National Conference on Artificial Intelligence*, edited by Y. Gil and R. Mooney (AAAI, Menio Park, CA, 2006), pp. 470–475.
[17] S. White and P. Smyth, *Proceedings of the 5th SIAM International Conference on Data Mining*, edited by H. Kargupta, J. Srivastava, C. Kamath, and A. Goodman (Society for Industrial and Applied Mathematics, Philadelphia, PA, 2005), pp. 274–284.
[18] R. Guimera and L. A. Nunes Amaral, Nature (London) **433**, 895 (2005).
[19] A. Clauset, M. E. J. Newman, and C. Moore, Phys. Rev. E **70**, 066111 (2004).
[20] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, and D. Wagner, e-print arXiv:physics/0608255.
[21] L. Danon, J. Duch, A. Diaz-Guilera, and A. Arenas, J. Stat. Mech.: Theory Exp. (2005) P09008.
[22] S. Fortunato and M. Barthelemy, Proc. Natl. Acad. Sci. U.S.A. **104**, 36 (2007).
[23] S. Muff, F. Rao, and A. Caflisch, Phys. Rev. E **72**, 056107 (2005).
[24] R. Guimerà, M. Sales-Pardo, and L. A. Nunes Amaral, Phys. Rev. E **70**, 025101(R) (2004).
[25] J. Ruan and W. Zhang, *Proceedings of the IEEE International Conference on Data Mining*, edited by N. Ramakrishnan, O. R. Zaïane, Y. Shi, C. W. Clifton, and X. Wu (IEEE Computer Society, Washington, D.C., 2007), pp. 643–648.
[26] J. Shi and J. Malik, IEEE Trans. Pattern Anal. Mach. Intell. **22**, 888 (2000).
[27] A. Y. Ng, M. I. Jordan, and Y. Weiss, *Advances in Neural Information Processing System*, edited by T. G. Dietterich, S. Becker, and Z. Ghahramani (MIT Press, Cambridge, MA, 2001), pp. 849–856.
[28] C. Elkan, *Proceedings of the 20th International Conference on Machine Learning*, edited by T. Fawcett, and N. Mishra (AAAI, Menlo Park, CA, 2003), pp. 147–153.
[29] http://cic.cs.wustl.edu/qcut/supplemental.pdf
[30] R. Milo, N. Kashtan, S. Itzkovitz, M. E. J. Newman, and U. Alon, e-print arXiv:cond-mat/0312028.
[31] J. Duch and A. Arenas, Phys. Rev. E **72**, 027104 (2005).
[32] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining (Adaptive Computation and Machine Learning)* (MIT, Cambridge, MA, 2001).
[33] L. Danon, A. Diaz-Guilera, and A. Arenas, J. Stat. Mech.: Theory Exp. (2006) P11010.
[34] A. Arenas, A. Diaz-Guilera, and C. Perez-Vicente, Phys. Rev. Lett. **96**, 114102 (2006).
[35] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, Nature (London) **435**, 814 (2005).
[36] J. Reichardt and S. Bornholdt, Phys. Rev. Lett. **93**, 218701 (2004).
[37] P. N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining* (Addison-Wesley, Reading, MA, 2005).
[38] E. B. Fowlkes and C. L. Mallows, J. Am. Stat. Assoc. **78**, 553 (1983).
[39] M. Meila, J. Multivariate Anal. **98**, 873 (2007).
[40] N. J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, and A. P. Datta, Nature (London) **440**, 637 (2006).
[41] H. W. Mewes, D. Frishman, K. F. Mayer, M. Munsterkotter, O. Noubibou, P. Pagel, T. Rattei, M. Oesterheld, A. Ruepp, and V. Stumpflen, Nucleic Acids Res. **34**, D169 (2006).