Near linear time algorithm to detect community structures in large-scale networks

Usha Nandini Raghavan,[1] Réka Albert,[2] and Soundar Kumara[1]
[1]*Department of Industrial Engineering, The Pennsylvania State University, University Park, Pennsylvania 16802, USA*
[2]*Department of Physics, The Pennsylvania State University, University Park, Pennsylvania 16802, USA*

(Received 9 April 2007; published 11 September 2007)

Community detection and analysis is an important methodology for understanding the organization of various real-world networks and has applications in problems as diverse as consensus formation in social communities or the identification of functional modules in biochemical networks. Currently used algorithms that identify the community structures in large-scale real-world networks require *a priori* information such as the number and sizes of communities or are computationally expensive. In this paper we investigate a simple label propagation algorithm that uses the network structure alone as its guide and requires neither optimization of a predefined objective function nor prior information about the communities. In our algorithm every node is initialized with a unique label and at every step each node adopts the label that most of its neighbors currently have. In this iterative process densely connected groups of nodes form a consensus on a unique label to form communities. We validate the algorithm by applying it to networks whose community structures are known. We also demonstrate that the algorithm takes an almost linear time and hence it is computationally less expensive than what was possible so far.

DOI: 10.1103/PhysRevE.76.036106 PACS number(s): 89.75.Fb, 89.75.Hc, 87.23.Ge, 02.10.Ox

I. INTRODUCTION

A wide variety of complex systems can be represented as networks. For example, the World Wide Web (WWW) is a network of web pages interconnected by hyperlinks; social networks are represented by people as nodes and their relationships by edges; and biological networks are usually represented by biochemical molecules as nodes and the reactions between them by edges. Most of the research in the recent past focused on understanding the evolution and organization of such networks and the effect of network topology on the dynamics and behaviors of the system [1–4]. Finding community structures in networks is another step toward understanding the complex systems they represent.

A community in a network is a group of nodes that are similar to each other and dissimilar from the rest of the network. It is usually thought of as a group where nodes are densely interconnected and sparsely connected to other parts of the network [4–6]. There is no universally accepted definition for a community, but it is well known that most real-world networks display community structures. There has been a lot of effort recently in defining, detecting, and identifying communities in real-world networks [5,7–15]. The goal of a community detection algorithm is to find groups of nodes of interest in a given network. For example, a community in the WWW network indicates a similarity among nodes in the group. Hence if we know the information provided by a small number of web pages, then it can be extrapolated to other web pages in the same community. Communities in social networks can provide insights about common characteristics or beliefs among people that makes them different from other communities. In biomolecular interaction networks, segregating nodes into functional modules can help identify the roles or functions of individual molecules [10]. Further, in many large-scale real-world networks, communities can have distinct properties which are lost in their combined analysis [1].

Community detection is similar to the well studied network partitioning problems [16–18]. The *network partitioning* problem is in general defined as the partitioning of a network into *c* (a fixed constant) groups of approximately equal sizes, minimizing the number of edges between groups. This problem is *NP*-hard and efficient heuristic methods have been developed over years to solve the problem [16–20]. Much of this work is motivated by engineering applications including very large scale integrated (VLSI) circuit layout designs and mapping of parallel computations. Thompson [21] showed that one of the important factors affecting the minimum layout area of a given circuit in a chip is its bisection width. Also, to enhance the performance of a computational algorithm, where nodes represent computations and edges represent communications, the nodes are divided equally among the processors so that the communications between them are minimized.

The goal of a network partitioning algorithm is to divide any given network into approximately equal size groups irrespective of node similarities. Community detection, on the other hand, finds groups that either have an inherent or an externally specified notion of similarity among nodes within groups. Furthermore, the number of communities in a network and their sizes are not known beforehand and they are established by the community detection algorithm.

Many algorithms have been proposed to find community structures in networks. Hierarchical methods divide networks into communities, successively, based on a dissimilarity measure, leading to a series of partitions from the entire network to singleton communities [5,15]. Similarly one can also successively group together smaller communities based on a similarity measure leading again to a series of partitions [22,23]. Due to the wide range of partitions, structural indices that measure the strength of community structures are used in determining the most relevant ones. Simulation based methods are also often used to find partitions with a strong community structure [10,24]. Spectral [17,25] and flow maximization (cut minimization) methods [9,26] have been successfully used in dividing networks into two or more communities.

In this paper, we propose a localized community detection algorithm based on label propagation. Each node is initialized with a unique label and at every iteration of the algorithm, each node adopts a label that a maximum number of its neighbors have, with ties broken uniformly randomly. As the labels propagate through the network in this manner, densely connected groups of nodes form a consensus on their labels. At the end of the algorithm, nodes having the same labels are grouped together as communities. As we will show, the advantage of this algorithm over the other methods is its simplicity and time efficiency. The algorithm uses the network structure to guide its progress and does not optimize any specific chosen measure of community strengths. Furthermore, the number of communities and their sizes are not known *a priori* and are determined at the end of the algorithm. We will show that the community structures obtained by applying the algorithm on previously considered networks, such as Zachary's karate club friendship network and the U.S. college football network, are in agreement with the actual communities present in these networks.

II. DEFINITIONS AND PREVIOUS WORK

As mentioned earlier, there is no unique definition of a community. One of the simplest definitions of a community is a clique, that is, a group of nodes where there is an edge between every pair of nodes. Cliques capture the intuitive notion of a community [6] where every node is related to every other node and hence have strong similarities with each other. An extension of this definition was used by Palla *et al.* in [14], who define a community as a chain of adjacent cliques. They define two k cliques (cliques on k nodes) to be adjacent if they share $k-1$ nodes. These definitions are strict in the sense that the absence of even one edge implies that a clique (and hence the community) no longer exists. k clans and k clubs are more relaxed definitions while still maintaining a high density of edges within communities [14]. A group of nodes is said to form a k clan if the shortest path length between any pair of nodes, or the diameter of the group, is at most k. Here the shortest path only uses the nodes within the group. A k club is defined similarly, except that the subnetwork induced by the group of nodes is a maximal subgraph of diameter k in the network.

Definitions based on degrees (number of edges) of nodes within the group relative to their degrees outside the group were given by Radicchi *et al.* [15]. If d_i^{in} and d_i^{out} are the degrees of node i within and outside of its group U, then U is said to form a *strong* community if $d_i^{\text{in}} > d_i^{\text{out}}, \quad \forall i \in U$. If $\Sigma_{i\in U} d_i^{\text{in}} > \Sigma_{i\in U} d_i^{\text{out}}$, then U is a community in the *weak* sense. Other definitions based on degrees of nodes can be found in [6].

There can exist many different partitions of nodes in the network that satisfy a given definition of community. In most cases [4,22,26–28], the groups of nodes found by a community detection algorithm are assumed to be communities irrespective of whether they satisfy a specific definition or not. To find the best community structures among them we need a measure that can quantify the strength of a community obtained. One of the ways to measure the strength of a community is by comparing the density of edges observed within the community with the density of edges in the network as a whole [6]. If the number of edges observed within a community U is e_U, then under the assumption that the edges in the network are uniformly distributed among pairs of nodes, we can calculate the probability P that the expected number of edges within U is larger than e_U. If P is small, then the observed density in the community is greater than the expected value. A similar definition was recently adopted by Newman [13], where the comparison is between the observed density of edges within communities and the expected density of edges within the same communities in randomized networks that nevertheless maintain every node's degree. This was termed the *modularity measure* Q, where $Q = \Sigma_i (e_{ii} - a_i^2), \quad \forall i$. e_{ii} is the observed fraction of edges within group i and a_i^2 is the expected fraction of edges within the same group i. Note that if e_{ij} is the fraction of edges in the network that run between group i and group j, then $a_i = \Sigma_j e_{ij}$. $Q=0$ implies that the density of edges within groups in a given partition is no more than what would be expected by a random chance. Q closer to 1 indicates stronger community structures.

Given a network with n nodes and m edges $N(n,m)$, any community detection algorithm finds subgroups of nodes. Let $C_1, C_2, \ldots, C_p$ be the communities found. In most algorithms, the communities found satisfy the following constraints: (i) $C_i \cap C_j = \emptyset$ for $i \neq j$ and (ii) $\cup_i C_i$ spans the node set in N.

A notable exception is Palla *et al.* [14] who define communities as a chain of adjacent k cliques and allow community overlaps. It takes exponential time to find all such communities in the network. They use these sets to study the overlapping structure of communities in social and biological networks. By forming another network where a community is represented by a node and edges between nodes indicates the presence of overlap, they show that such networks are also heterogeneous (fat-tailed) in their node degree distributions. Furthermore, if a community has overlapping regions with two other communities, then the neighboring communities are also highly likely to overlap.

The number of different partitions of a network $N(n,m)$ into just two disjoint subsets is 2^n and increases exponentially with n. Hence we need a quick way to find only relevant partitions. Girvan and Newman [5] proposed a divisive algorithm based on the concept of *edge betweenness centrality*, that is, the number of shortest paths among all pairs of nodes in the network passing through that edge. The main idea here is that edges that run between communities have higher betweenness values than those that lie within communities. By successively recalculating and removing edges with highest betweenness values, the network breaks down into disjoint connected components. The algorithm continues until all edges are removed from the network. Each step of the algorithm takes $O(mn)$ time and since there are m edges to be removed, the worst case running time is $O(m^2 n)$. As the algorithm proceeds one can construct a dendrogram (see Fig. 1) depicting the breaking down of the network into dis-

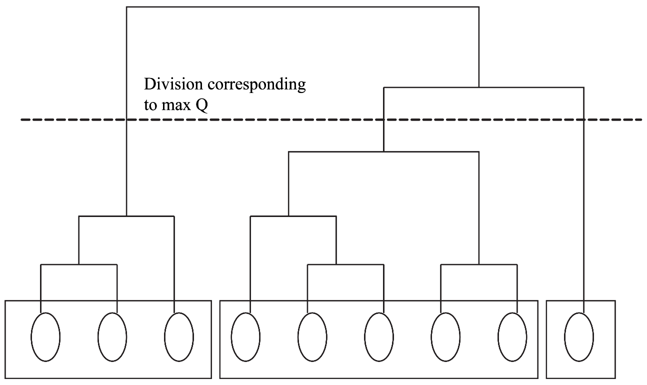

FIG. 1. An illustration of a dendrogram which is a tree representation of the order in which nodes are segregated into different groups or communities.

joint connected components. Hence for any given h such that $1 \leq h \leq n$, at most one partition of the network into h disjoint subgroups is found. All such partitions in the dendrogram are depicted, irrespective of whether or not the subgroups in each partition represent a community. Radicchi *et al.* [15] propose another divisive algorithm where the dendrograms are modified to reflect only those groups that satisfy a specific definition of a community. Further, instead of edge betweenness centrality, they use a local measure called *edge clustering coefficient* as a criterion for removing edges. The edge clustering coefficient is defined as the fraction of number of triangles a given edge participates in, to the total number of possible such triangles. The clustering coefficient of an edge is expected to be the least for those running between communities and hence the algorithm proceeds by removing edges with low clustering coefficients. The total running time of this divisive algorithm is $O\left(\frac{m^4}{n^2}\right)$.

Similarly one can also define a topological similarity between nodes and perform an agglomerative hierarchical clustering [23,29]. In this case, we begin with nodes in n different communities and group together communities that are the most similar. Newman [22] proposed an amalgamation method (similar to agglomerative methods) using the modularity measure Q, where at each step those two communities are grouped together that give rise to the maximum increase or smallest decrease in Q. This process can also be represented as a dendrogram and one can cut across the dendrogram to find the partition corresponding to the maximum value of Q (see Fig. 1). At each step of the algorithm one compares at most m pairs of groups and requires at most $O(n)$ time to update the Q value. The algorithm continues until all the n nodes are in one group and hence the worst case running time of the algorithm is $O[n(m+n)]$. The algorithm of Clauset *et al.* [30] is an adaptation of this agglomerative hierarchical method, but uses a clever data structure to store and retrieve information required to update Q. In effect, they reduce the time complexity of the algorithm to $O(md \log n)$, where d is the depth of the dendrogram obtained. In networks that have a hierarchical structure with communities at many scales, $d \sim \log n$. There have also been other heuristic and simulation based methods that find partitions of a given network maximizing the modularity measure Q [10,24].

Label flooding algorithms have also been used in detecting communities in networks [27,28]. In [27], the authors propose a local community detection method where a node is initialized with a label which then propagates step by step via the neighbors until it reaches the end of the community, where the number of edges proceeding outward from the community drops below a threshold value. After finding the local communities at all nodes in the network, an $n \times n$ matrix is formed, where the ijth entry is 1 if node j belongs to the community started from i and 0 otherwise. The rows of the matrix are then rearranged such that the similar ones are closer to each other. Then, starting from the first row they successively include all the rows into a community until the distance between two successive rows is large and above a threshold value. After this a new community is formed and the process is continued. Forming the rows of the matrix and rearranging them requires $O(n^3)$ time and hence the algorithm is time-consuming.

Wu and Huberman [26] propose a linear time $[O(m+n)]$ algorithm that can divide a given network into two communities. Suppose that one can find two nodes (x and y) that belong to two different communities, then they are initialized with values 1 and 0, respectively. All other nodes are initialized with value 0. Then at each step of the algorithm, all nodes (except x and y) update their values as follows. If $z_1, z_2, \ldots, z_k$ are neighbors of a node z, then the value V_z is updated as $\frac{V_{z_1}+V_{z_2}+\cdots+V_{z_k}}{k}$. This process continues until convergence. The authors show that the iterative procedure converges to a unique value, and the convergence of the algorithm does not depend on the size n of the network. Once the required convergence is obtained, the values are sorted between 0 and 1. Going through the spectrum of values in descending order, there will be a sudden drop at the border of two communities. This gap is used in identifying the two communities in the network. A similar approach was used by Flake *et al.* [9] to find the communities in the WWW network. Here, given a small set of nodes (source nodes), they form a network of web pages that are within a bounded distance from the sources. Then by designating (or artificially introducing) sink nodes, they solve for the maximum flow from the sources to the sinks. In doing so one can then find the *minimum cut* corresponding to the maximum flow. The connected component of the network containing the source nodes after the removal of the cut set is then the required community.

Spectral bisection methods [25] have been used extensively to divide a network into two groups so that the number of edges between groups is minimized. Eigenvectors of the Laplacian matrix (L) of a given network are used in the bisection process. It can be shown that L has only real nonnegative eigenvalues ($0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$) and minimizing the number of edges between groups is the same as minimizing the positive linear combination $M = \Sigma_i s_i^2 \lambda_i$, where $s_i = u_i^T z$ and u_i is the eigenvector of L corresponding to λ_i. z is the decision vector whose ith entry can be either 1 or -1 denoting to which of the two groups node i belongs. To minimize M, z is chosen as parallel as possible to the eigenvector corresponding to the second smallest eigenvalue. (The smallest eigenvalue is 0 and choosing z parallel to the corre-

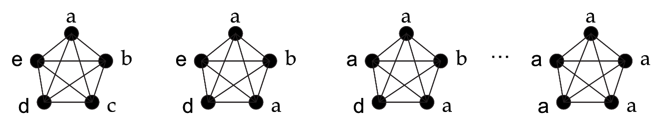

FIG. 2. Nodes are updated one by one as we move from left to right. Due to a high density of edges (highest possible in this case), all nodes acquire the same label.

sponding eigenvector gives a trivial solution.) This bisection method has been extended to finding communities in networks that maximize the modularity measure Q [25]. Q can be written as a positive linear combination of eigenvalues of the matrix B, where B is defined as the difference of the two matrices A and P. A_{ij} is the observed number of edges between nodes i and j and P_{ij} is the expected number of edges between i and j if the edges fall randomly between nodes, while maintaining the degree of each node. Since Q has to be maximized, z is chosen as parallel as possible to the eigenvector corresponding to the largest eigenvalue.

Since many real-world complex networks are large in size, time efficiency of the community detection algorithm is an important consideration. When no *a priori* information is available about the likely communities in a given network, finding partitions that optimize a chosen measure of community strength is normally used. Our goal in this paper is to develop a simple time-efficient algorithm that requires no prior information (such as number, sizes, or central nodes of the communities) and uses only the network structure to guide the community detection. The proposed mechanism for such an algorithm which does not optimize any specific measure or function is detailed in the following section.

III. COMMUNITY DETECTION USING LABEL PROPAGATION

The main idea behind our label propagation algorithm is the following. Suppose that a node x has neighbors $x_1, x_2, \ldots, x_k$ and that each neighbor carries a label denoting the community to which they belong. Then x determines its community based on the labels of its neighbors. We assume that each node in the network chooses to join the community to which the maximum number of its neighbors belong, with ties broken uniformly randomly. We initialize every node with unique labels and let the labels propagate through the network. As the labels propagate, densely connected groups of nodes quickly reach a consensus on a unique label (see Fig. 2). When many such dense (consensus) groups are created throughout the network, they continue to expand outwards until it is possible to do so. At the end of the propagation process, nodes having the same labels are grouped together as one community.

We perform this process iteratively, where at every step, each node updates its label based on the labels of its neighbors. The updating process can either be synchronous or asynchronous. In synchronous updating, node x at the tth iteration updates its label based on the labels of its neighbors at iteration $t-1$. Hence $C_x(t)=f(C_{x_1}(t-1),\ldots,C_{x_k}(t-1))$, where $c_x(t)$ is the label of node x at time t. The problem,

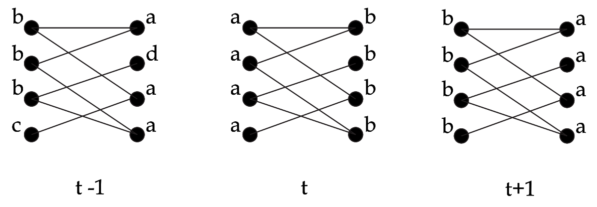

FIG. 3. An example of a bi-partite network in which the label sets of the two parts are disjoint. In this case, due to the choices made by the nodes at step t, the labels on the nodes oscillate between a and b.

however, is that subgraphs in the network that are bipartite or nearly bipartite in structure lead to oscillations of labels (see Fig. 3). This is especially true in cases where communities take the form of a star graph. Hence we use asynchronous updating where $C_x(t)=f(C_{x_{i1}}(t),\ldots,C_{x_{im}}(t),C_{x_{i(m+1)}}(t-1),\ldots,C_{x_{ik}}(t-1))$ and $x_{i1},\ldots,x_{im}$ are neighbors of x that have already been updated in the current iteration while $x_{i(m+1)},\ldots,x_{ik}$ are neighbors that are not yet updated in the current iteration. The order in which all the n nodes in the network are updated at each iteration is chosen randomly. Note that while we have n different labels at the beginning of the algorithm, the number of labels reduces over iterations, resulting in only as many unique labels as there are communities.

Ideally the iterative process should continue until no node in the network changes its label. However, there could be nodes in the network that have an equal maximum number of neighbors in two or more communities. Since we break ties randomly among the possible candidates, the labels on such nodes could change over iterations even if the labels of their neighbors remain constant. Hence we perform the iterative process until every node in the network has a label to which the maximum number of its neighbors belongs. By doing so we obtain a partition of the network into disjoint communities, where every node has at least as many neighbors within its community as it has with any other community. If $C_1,\ldots,C_p$ are the labels that are currently active in the network and $d_i^{C_j}$ is the number of neighbors node i has with nodes of label C_j, then the algorithm is stopped when for every node i,

$$\text{If } i \text{ has label } C_m \text{ then } d_i^{C_m} \geq d_i^{C_j} \quad \forall\, j.$$

At the end of the iterative process nodes with the same label are grouped together as communities. Our stop criterion characterizing the obtained communities is similar (but not identical) to the definition of *strong communities* proposed by Radicchi *et al.* [15]. While strong communities require each node to have strictly more neighbors within its community than outside, the communities obtained by the label propagation process require each node to have at least as many neighbors within its community as it has with each of the other communities. We can describe our proposed label propagation algorithm in the following steps.

(i) Initialize the labels at all nodes in the network. For a given node x, $C_x(0)=x$.

(ii) Set $t=1$.

(iii) Arrange the nodes in the network in a random order and set it to X.

(iv) For each $x \in X$ chosen in that specific order, let $C_x(t)=f(C_{x_{i1}}(t),\ldots,C_{x_{im}}(t),C_{x_{i(m+1)}}(t-1),\ldots,C_{x_{ik}}(t-1))$. f here returns the label occurring with the highest frequency among neighbors and ties are broken uniformly randomly.

(v) If every node has a label that the maximum number of their neighbors have, then stop the algorithm. Else, set $t=t+1$ and go to (iii).

Since we begin the algorithm with each node carrying a unique label, the first few iterations result in various small pockets (dense regions) of nodes forming a consensus (acquiring the same label). These consensus groups then gain momentum and try to acquire more nodes to strengthen the group. However, when a consensus group reaches the border of another consensus group, they start to compete for members. The within-group interactions of the nodes can counteract the pressures from outside if there are less between-group edges than within-group edges. The algorithm converges, and the final communities are identified, when a global consensus among groups is reached. Note that even though the network as one single community satisfies the stop criterion, this process of group formation and competition discourages all nodes from acquiring the same label in the case of heterogeneous networks with an underlying community structure. In the case of homogeneous networks such as Erdős-Rényi random graphs [31] that do not have community structures, the label propagation algorithm identifies the giant connected component of these graphs as a single community.

Our stop criterion is only a condition and not a measure that is being maximized or minimized. Consequently there is no unique solution and more than one distinct partition of a network into groups satisfies the stop criterion (see Figs. 4 and 5). Since the algorithm breaks ties uniformly randomly, early on in the iterative process when possibilities of ties are high, a node may vote in favor of a randomly chosen community. As a result, multiple community structures are reachable from the same initial condition.

If we know the set of nodes in the network that are likely to act as centers of attraction for their respective communities, then it would be sufficient to initialize such nodes with unique labels, leaving the remaining nodes unlabeled. In this case when we apply the proposed algorithm the unlabeled nodes will have a tendency to acquire labels from their closest attractor and join that community. Also, restricting the set of nodes initialized with labels will reduce the range of possible solutions that the algorithm can produce. Since it is generally difficult to identify nodes that are central to a community before identifying the community itself, here we give all nodes equal importance at the beginning of the algorithm and provide them each with unique labels.

We apply our algorithm to the following networks. The first one is Zachary's karate club network which is a network of friendship among 34 members of a karate club [32]. Over a period of time the club split into two factions due to leadership issues and each member joined one of the two factions. The second network that we consider is the U.S. college football network that consists of 115 college teams represented as nodes and has edges between teams that played each other during the regular season in the year 2000 [5]. The teams are divided into conferences (communities) and each team plays more games within its own conference than interconference games. Next is the coauthorship network of 16 726 scientists who have posted preprints on the condensed matter archive at www.arxiv.org; the edges connect scientists who coauthored a paper [33]. It has been

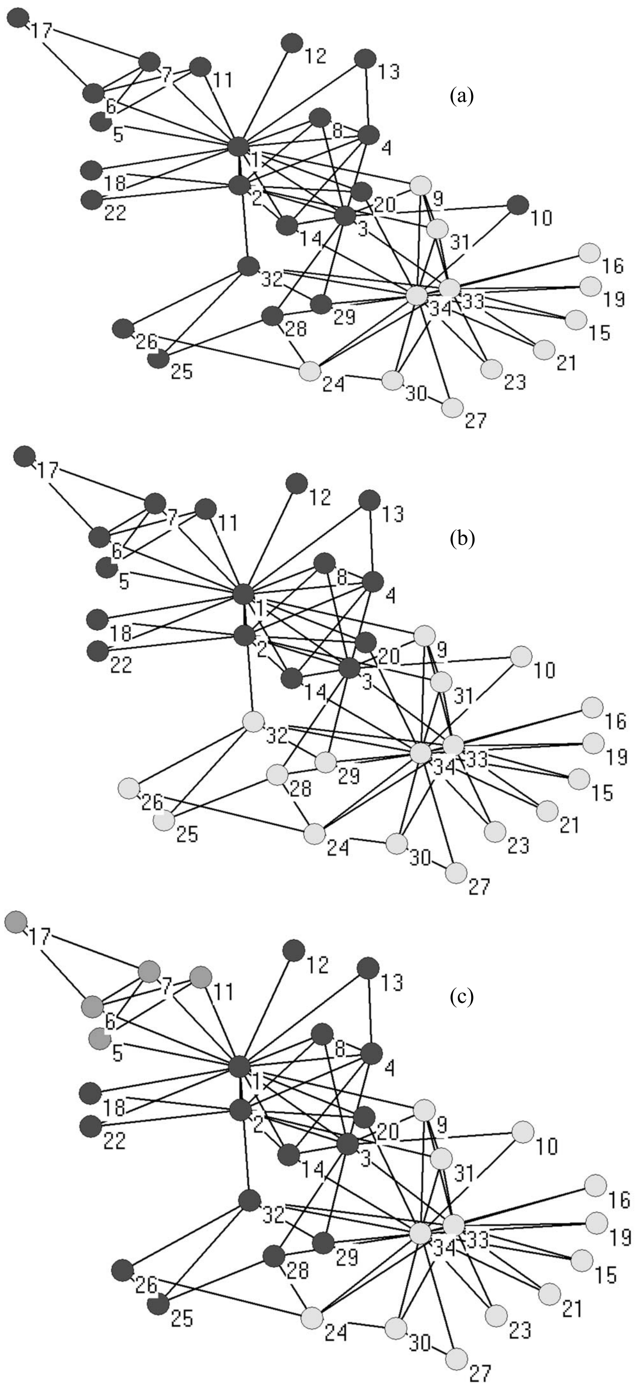

FIG. 4. (a)–(c) are three different community structures identified by the algorithm on Zachary's karate club network. The communities can be identified by their shades of gray colors.

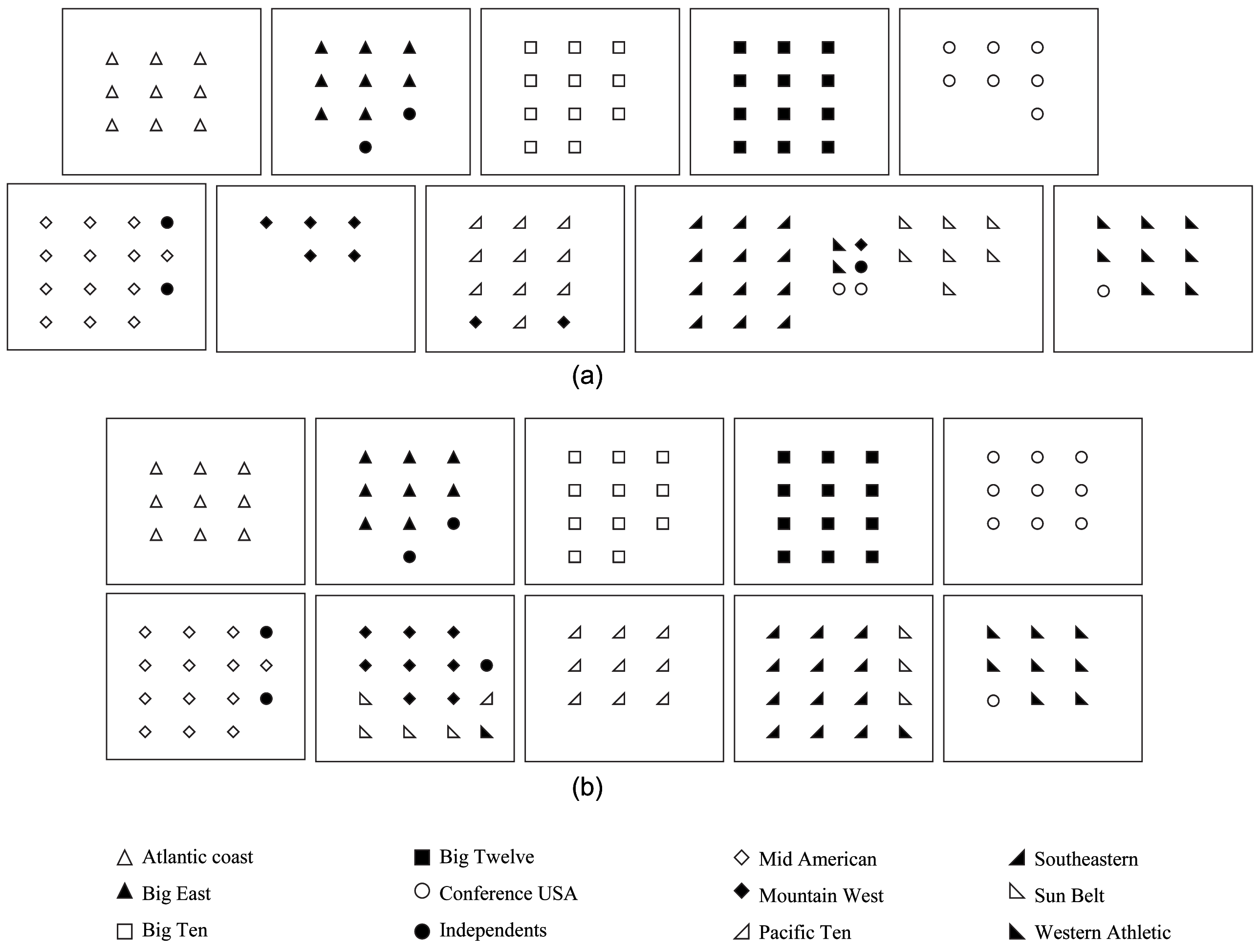

FIG. 5. The grouping of U.S. college football teams into conferences are shown in (a) and (b). Each solution [(a) and (b)] is an aggregate of five different solutions obtained by applying the algorithm on the college football network.

shown that communities in coauthorship networks are made up by researchers working in the same field or are research groups [22]. Along similar lines one can expect an actor collaboration network to have communities containing actors of a similar genre. Here we consider an actor collaboration network of 374 511 nodes and edges running between actors who have acted in at least one movie together [3]. We also consider a protein-protein interaction network [34] consisting of 2115 nodes. The communities are likely to reflect functional groupings of this network. And finally we consider a subset of the WWW) consisting of 325 729 web pages within the nd.edu domain and hyperlinks interconnecting them [2]. Communities here are expected to be groups of pages on similar topics.

A. Multiple community structures

Figure 4 shows three different solutions obtained for the Zachary's karate club network and Fig. 5 shows two different solutions obtained for the U.S. college football network. We will show that even though we obtain different solutions (community structure), they are similar to each other. To find the percentage of nodes classified in the same group in two different solutions, we form a matrix M, where M_{ij} is the number of nodes common to community i in one solution and community j in the other solution. Then we calculate $f_{\text{same}}=\frac{1}{2}(\Sigma_i \max_j\{M_{ij}\}+\Sigma_j \max_i\{M_{ij}\})\frac{100}{n}$. Given a network whose communities are already known, a community detection algorithm is commonly evaluated based on the percentage (or number) of nodes that are grouped into the correct communities [22,26]. f_{same} is similar, whereby fixing one solution we evaluate how close the other solution is to the fixed one and vice versa. While f_{same} can identify how close one solution is to another, it is, however, not sensitive to the seriousness of errors. For example, when few nodes from several different communities in one solution are fused together as a single community in another solution, the value of f_{same} does not change much. Hence we also use Jaccard's index which has been shown to be more sensitive to such differences between solutions [35]. If a stands for the pairs of nodes that are classified in the same community in both solutions, b for pairs of nodes that are in the same community in the first solution and different in the second, and c vice versa, then Jaccard's index is defined as $\frac{a}{a+b+c}$. It takes values between 0 and 1, with higher values indicating stronger similarity between the two solutions. Figure 6 shows the similarities between solutions obtained from applying the algorithm five different times on the same network. For a

--	83.8235	86.7647	97.0588	92.6470
0.5545	--	91.1764	86.7647	91.1764
0.6274	0.7084	--	89.7058	94.1176
0.8813	0.6098	0.6955	--	89.7058
0.8188	0.7102	0.7908	0.7272	--

Karate club friendship network

Q : 0.355 – 0.399

--	91.3043	92.1739	90	94.7826
0.7294	--	89.5652	84.7826	89.5652
0.6971	0.6223	--	83.9130	88.6956
0.6572	0.5903	0.5213	--	95.2173
0.7586	0.6787	0.5871	0.8477	--

US College football network

Q : 0.457 – 0.476

--	86.5248	85.8156	85.7210	85.3901
0.8569	--	86.2884	86.4302	85.7683
0.8316	0.8398	--	86.0520	83.9007
0.8430	0.8551	0.8373	--	85.4137
0.7998	0.8159	0.7869	0.8124	--

Protein-protein interaction network

Q : 0.738 – 0.745

--	84.6735	84.7213	83.9680	85.1488
0.6019	--	84.4553	84.8050	85.7683
0.6082	0.5985	--	84.5958	84.5689
0.5909	0.6082	0.5974	--	85.8543
0.6036	0.6042	0.6036	0.6254	--

Co-authorship network

Q : 0.720 – 0.722

--	91.9743	91.9000	92.2589	92.0706
0.5431	--	92.0527	92.2535	92.3695
0.5358	0.5013	--	91.9867	92.3490
0.5454	0.5576	0.4883	--	92.4904
0.5425	0.5921	0.5056	0.5703	--

World Wide Web

Q : 0.857 – 0.864

--	94.5946	91.8218
0.8284	--	92.2034
0.6026	0.6810	--

Actor collaboration

Q : 0.437 – 0.528

FIG. 6. Similarities between five different solutions obtained for each network is tabulated. An entry in the ith row and jth column in the lower triangle of each of the tables is the Jaccard's similarity index for solutions i and j of the corresponding network. Entries in the ith row and jth column in the upper triangle of the tables are the values of the measure f_{same} for solutions i and j in the respective networks. The range of modularity values Q obtained for the five different solutions is also given for each network.

given network, the ijth entry in the lower triangle of the table is the Jaccard index for solutions i and j, while the ijth entry in the upper triangle is the measure f_{same} for solutions i and j. We can see that the solutions obtained from the five different runs are similar, implying that the proposed label propagation algorithm can effectively identify the community structure of any given network. Moreover, the tight range and high values of the modularity measure Q obtained for the five solutions (Fig. 6) suggest that the partitions denote significant community structures.

B. Aggregate

It is difficult to pick one solution as the best among several different ones. Furthermore, one solution may be able to identify a community that was not discovered in the other and vice versa. Hence an aggregate of all the different solutions can provide a community structure containing the most useful information. In our case a solution is a set of labels on the nodes in the network and all nodes having the same label form a community. Given two different solutions, we combine them as follows; let C^1 denote the labels on the nodes in solution 1 and C^2 denote the labels on the nodes in solution 2. Then, for a given node x, we define a new label as $C_x = (C_x^1, C_x^2)$ (see Fig. 7). Starting with a network initialized with labels C we perform the iterative process of label propagation until every node in the network is in a community to which the maximum number of its neighbors belongs. As and when new solutions are available they are combined one by one with the aggregate solution to form a new aggregate solution. Note that when we aggregate two solutions, if a community T in one solution is broken into two (or more) different communities S_1 and S_2 in the other, then by defining the new labels as described above we are showing preferences to the smaller communities S_1 and S_2 over T. This is only one of the many ways in which different solutions can be aggregated. For other methods of aggregation used in community detection refer to [26,36,37].

Figure 8 shows the similarities between aggregate solutions. The algorithm was applied on each network 30 times and the solutions were recorded. An ijth entry is the Jaccard index for the aggregate of the first $5i$ solutions with the ag-

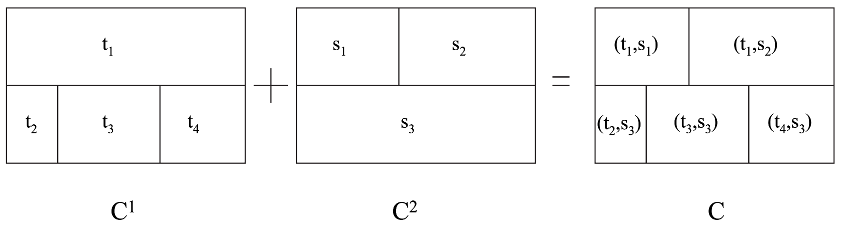

FIG. 7. An example of aggregating two community structure solutions. t_1, t_2, t_3, and t_4 are labels on the nodes in a network obtained from solution 1 and denoted as C^1. The network is partitioned into groups of nodes having the same labels. s_1, s_2, and s_3 are labels on the nodes in the same network obtained from solution 2 and denoted as C^2. All nodes that had label t_1 in solution 1 are split into two groups with each group having labels s_1 and s_2, respectively, while all nodes with labels t_3, t_4, or t_5 in solution 1 have labels s_3 in solution 2. C represents the new labels defined from C^1 and C^2.

gregate of the first $5j$ solutions. We observe that the aggregate solutions are very similar in nature and hence a small set of solutions (5 in this case) can offer as much insight about the community structure of a network as can a larger solution set. In particular, the WWW network which had low similarities between individual solutions (Jaccard index range 0.4883–0.5931), shows considerably improved similarities (Jaccard index range 0.6604–0.7196) between aggregate solutions.

IV. VALIDATION OF THE COMMUNITY DETECTION ALGORITHM

Since we know the communities present in Zachary's karate club and the U.S. football network, we explicitly verify the accuracy of the algorithm by applying it on these networks. We find that the algorithm can effectively unearth the underlying community structures in the respective networks. The community structures obtained by using our algorithm on Zachary's karate club network is shown in Fig. 4. While all three solutions are outcomes of the algorithm applied to the network, Fig. 4(b) reflects the true solution [32].

Figure 5 gives two solutions for the U.S. college football network. The algorithm was applied to this network ten different times and the two solutions are the aggregate of the first five and remaining five solutions. In both Figs. 5(a) and 5(b), we can see that the algorithm can effectively identify all the conferences with the exception of Sunbelt. The reason for the discrepancy is the following: among the seven teams

US college football network

--	0.7455	0.7713	0.7504	0.8851	0.7053
0.7805	--	0.9277	0.6853	0.7585	0.6417
0.8777	0.8814	--	0.6867	0.7817	0.6508
0.8777	0.8814	1	--	0.8256	0.6512
0.8777	0.8814	1	1	--	0.7888
0.7805	1	0.8814	0.8814	0.8814	--

Karate club friendship network

Co-authorship network

--	0.7691	0.7291	0.7368	0.7349	0.7578
0.8926	--	0.7560	0.7561	0.7597	0.7722
0.8927	0.8827	--	0.7360	0.7322	0.7604
0.9002	0.8887	0.8942	--	0.7717	0.7712
0.9003	0.8803	0.8885	0.8966	--	0.7642
0.9011	0.8864	0.8852	0.9062	0.8966	--

Protein-protein interaction network

--		
0.6545	--	
0.7196	0.6604	--

World Wide Web

FIG. 8. Similarities between aggregate solutions obtained for each network. An entry in the ith row and jth column in the tables is Jaccard's similarity index between the aggregate of the first $5i$ and the first $5j$ solutions. While similarities between solutions for the karate club friendship network and the protein-protein interaction network are represented in the lower triangles of the first two tables, the entries in the upper triangle of these two tables are for the U.S. college football network and the coauthorship network, respectively. The similarities between aggregate solutions for the WWW is given in the lower triangle of the third table.

in the Sunbelt conference, four teams ($\text{Sunbelt}_4 = \{$North-Texas, Arkansas State, Idaho, New Mexico State$\}$ have all played each other and three teams ($\text{Sunbelt}_3 = \{$Louisiana-Monroe, Middle-Tennessee State, Louisiana-Lafayette$\}$) have again played one another. There is only one game connecting Sunbelt_4 and Sunbelt_3, namely, the game between North-Texas and Louisiana-Lafayette. However, four teams from the Sunbelt conference (two each from Sunbelt_4 and Sunbelt_3) have together played with seven different teams in the Southeastern conference. Hence we have the Sunbelt conference grouped together with the Southeastern conference in Fig. 5(a). In Fig. 5(b), the Sunbelt conference breaks into two, with Sunbelt_3 grouped together with Southeastern and Sunbelt_4 grouped with an independent team (Utah State), a team from Western Atlantic (Boise State), and the Mountain West conference. The latter grouping is due to the fact that every member of Sunbelt_4 has played with Utah State and with Boise State, who have together played five games with four different teams in Mountain West. There are also five independent teams which do not belong to any specific conference and are hence assigned by the algorithm to a conference where they have played the maximum number of their games.

V. TIME COMPLEXITY

It takes a near-linear time for the algorithm to run to its completion. Initializing every node with unique labels requires $O(n)$ time. Each iteration of the label propagation algorithm takes linear time in the number of edges $[O(m)]$. At each node x, we first group the neighbors according to their labels $[O(d_x)]$. We then pick the group of maximum size and assign its label to x, requiring a worst-case time of $O(d_x)$. This process is repeated at all nodes and hence an overall time is $O(m)$ for each iteration.

As the number of iterations increases, the number of nodes that are classified correctly increases. Here we assume that a node is classified correctly if it has a label that the maximum number of its neighbors have. From our experiments, we found that irrespective of n, 95% of the nodes or more are classified correctly by the end of iteration 5. Even in the case of Erdős-Rényi random graphs [31] with n between 100 and 10 000 and average degree 4, which do not have community structures, by iteration 5, 95% of the nodes or more are classified correctly. In this case, the algorithm identified all nodes in the giant connected component as belonging to one community.

When the algorithm terminates it is possible that two or more disconnected groups of nodes have the same label (the groups are connected in the network via other nodes of different labels). This happens when two or more neighbors of a node receive its label and pass the labels in different directions, which ultimately leads to different communities adopting the same label. In such cases, after the algorithm terminates one can run a simple breadth-first search on the subnetworks of each individual group to separate the disconnected communities. This requires an overall time of $O(m+n)$. When aggregating solutions, however, we rarely find disconnected groups within communities.

VI. DISCUSSION AND CONCLUSIONS

The proposed label propagation process uses only the network structure to guide its progress and requires no external parameter settings. Each node makes its own decision regarding the community to which it belongs based on the communities of its immediate neighbors. These localized decisions lead to the emergence of community structures in a given network. We verified the accuracy of community structures found by the algorithm using Zachary's karate club and the U.S. college football networks. Furthermore, the modularity measure Q was significant for all the solutions obtained, indicating the effectiveness of the algorithm. Each iteration takes a linear time $O(m)$, and although one can observe the algorithm beginning to converge significantly after about five iterations, the mathematical convergence is hard to prove. Other algorithms that run in a similar time scale include the algorithm of Wu and Huberman [26] [with time complexity $O(m+n)$] and that of Clauset *et al.* [30] which has a running time of $O(n \log^2 n)$.

The algorithm of Wu and Huberman is used to break a given network into only two communities. In this iterative process two chosen nodes are initialized with scalar values 1 and 0 and every node updates its value as the average of the values of its neighbors. At convergence, if a maximum number of a node's neighbors have values above a given threshold then so will the node. Hence a node tends to be classified to a community to which the maximum number of its neighbors belong. Similarly if in our algorithm we choose the same two nodes and provide them with two distinct labels (leaving the others unlabeled), the label propagation process will yield similar communities as the Wu and Huberman algorithm. However, to find more than two communities in the network, the Wu and Huberman algorithm needs to know *a priori* how many communities there are in the network. Furthermore, if one knows that there are c communities in the network, the algorithm proposed by Wu and Huberman can only find communities that are approximately of the same size, that is, $\frac{n}{c}$, and it is not possible to find communities with heterogeneous sizes. The main advantage of our proposed label propagation algorithm over the Wu and Huberman algorithm is that we do not need *a priori* information on the number and sizes of the communities in a given network; indeed such information usually is not available for real-world networks. Also, our algorithm does not make restrictions on the community sizes. It determines such information about the communities by using the network structure alone.

In our test networks, the label propagation algorithm found communities whose sizes follow approximately a power-law distribution $P(S>s) \sim s^{-\nu}$ with the exponent ν ranging between 0.5 and 2 (Fig. 9). This implies that there is no characteristic community size in the networks and it is consistent with previous observations [22,30,38]. While the community size distributions for the WWW and coauthorship networks approximately follow power laws with a cutoff, with exponents 1.15 and 1.98, respectively, there is a clear crossover from one scaling relation to another for the

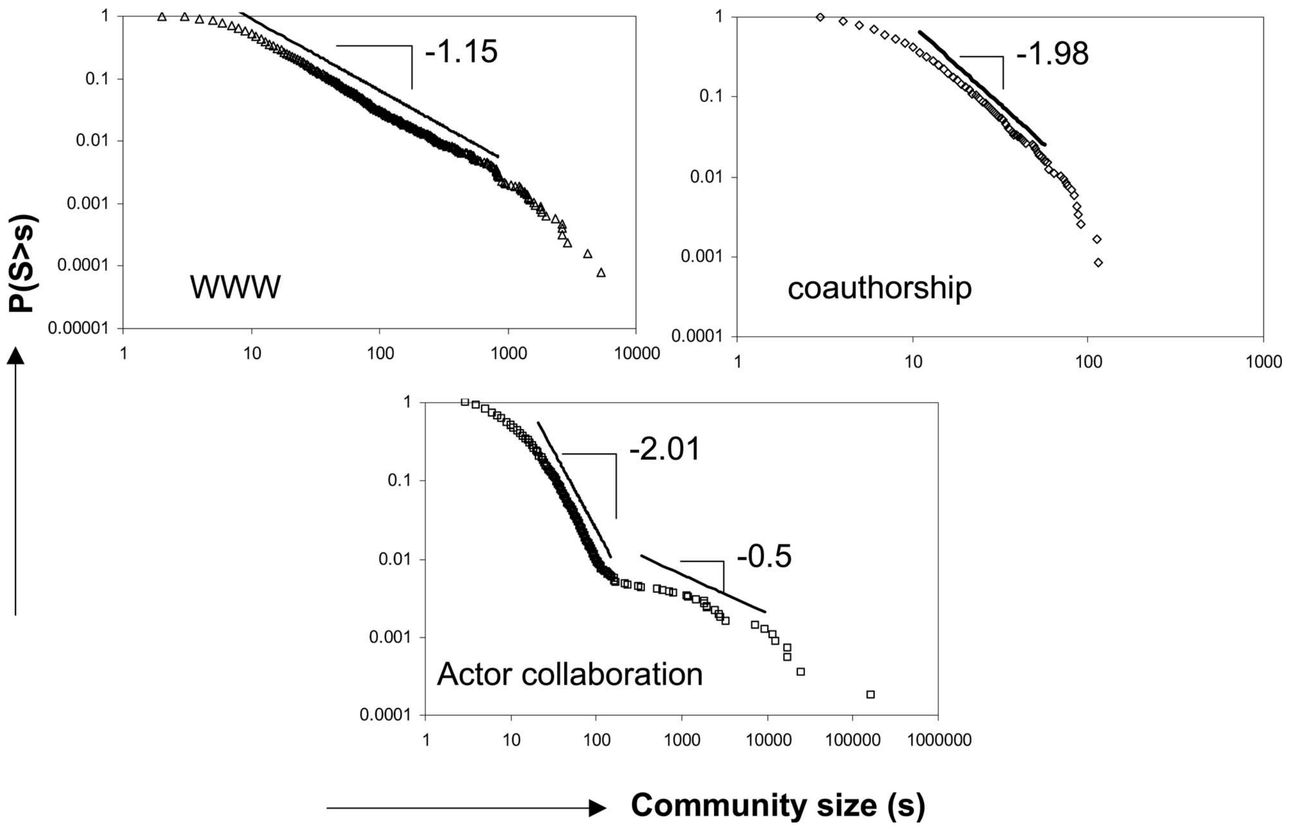

FIG. 9. The cumulative probability distributions of community sizes (s) are shown for the WWW, coauthorship and actor collaboration networks. They approximately follow power laws with the exponents as shown.

actor collaboration network. The community size distribution for the actor collaboration network has a power-law exponent of 2 for sizes up to 164 nodes and 0.5 between 164 and 7425 nodes (see Fig. 9).

In the hierarchical agglomerative algorithm of Clauset *et al.* [30], the partition that corresponds to the maximum Q is taken to be the most indicative of the community structure in the network. Other partitions with high Q values will have a structure similar to that of the maximum Q partition, as these solutions are obtained by progressively aggregating two groups at a time. Our proposed label propagation algorithm, on the other hand, finds multiple significantly modular solutions that have some amount of dissimilarity. For the WWW network in particular, the similarity between five different solutions is low, with the Jaccard index ranging between 0.4883 and 0.5921, yet all five are significantly modular with Q between 0.857 and 0.864. This implies that the proposed algorithm can find not just one but multiple significant community structures, supporting the existence of overlapping communities in many real-world networks [14].

ACKNOWLEDGMENTS

The authors would like to acknowledge the National Science Foundation (Grants No. SST 0427840, No. DMI 0537992, and No. CCF 0643529). One of the authors (R.A.) acknowledges support from the Sloan Foundation.

[1] R. Albert and A.-L. Barabási, Rev. Mod. Phys. **74**, 47 (2002).
[2] R. Albert, H. Jeong, and A.-L. Barabási, Nature (London) **401**, 130 (1999).
[3] A.-L. Barabási and R. Albert, Science **286**, 509 (1999).
[4] M. Newman, SIAM (Soc. Ind. Appl. Math.) Rev. **45**, 167 (2003).
[5] M. Girvan and M. Newman, Proc. Natl. Acad. Sci. U.S.A. **99**, 7821 (2002).
[6] S. Wasserman and K. Faust, *Social Network Analysis* (Cambridge University Press, Cambridge, England, 1994).
[7] L. Danon, A. Díaz-Guilera, and A. Arenas, J. Stat. Mech.: Theor. Exp. **2006** P11010 (2006).
[8] J. Eckmann and E. Moses, Proc. Natl. Acad. Sci. U.S.A. **99**, 5825 (2002).
[9] G. Flake, S. Lawrence, and C. Giles, Proceedings of the 6th ACM SIGKDD, 2000, pp. 150–160.
[10] R. Guimerà and L. Amaral, Nature (London) **433**, 895 (2005).
[11] M. Gustafsson, M. Hornquist, and A. Lombardi, Physica A **367**, 559 (2006).
[12] M. B. Hastings, Phys. Rev. E **74**, 035102(R) (2006).

[13] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
[14] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Nature (London) **435**, 814 (2005).
[15] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, Proc. Natl. Acad. Sci. U.S.A. **101**, 2658 (2004).
[16] D. Karger, J. ACM **47**, 46 (2000).
[17] B. Kernighan and S. Lin, Bell Syst. Tech. J. **29**, 291 (1970).
[18] C. Fiduccia and R. Mattheyses, Proceedings of the 19th Annual ACM IEEE Design Automation Conference, 1982, pp. 175–181.
[19] B. Hendrickson and R. Leland, SIAM (Soc. Ind. Appl. Math.) J. Sci. Comput. **16**, 452 (1995).
[20] M. Stoer and F. Wagner, J. ACM **44**, 585 (1997).
[21] C. Thompson, Proceedings of the 11th Annual ACM Symposium on Theory of Computing, 1979, pp. 81–88.
[22] M. E. J.Newman, Phys. Rev. E **69**, 066133 (2004).
[23] P. Pons and M. Latapy, e-print arXiv:physics/0512106.
[24] J. Duch and A. Arenas, Phys. Rev. E **72**, 027104 (2005).
[25] M. E. J. Newman, Phys. Rev. E **74**, 036104 (2006).
[26] F. Wu and B. Huberman, Eur. Phys. J. B **38**, 331 (2004).
[27] J. P. Bagrow and E. Bollt, Phys. Rev. E **72**, 046108 (2005).
[28] L. Costa, e-print arXiv:cond-mat/0405022.
[29] M. E. J. Newman, Eur. Phys. J. B **38**, 321 (2004).
[30] A. Clauset, M. E. J. Newman, and C. Moore, Phys. Rev. E **70**, 066111 (2004).
[31] B. Bollobás, *Random Graphs* (Academic Press, Orlando, FL, 1985).
[32] W. Zachary, J. Anthropol. Res. **33**, 452 (1977).
[33] M. Newman, Proc. Natl. Acad. Sci. U.S.A. **98**, 404 (2001).
[34] H. Jeong, S. Mason, A.-L. Barabási, and Z. Oltvai, Nature (London) **411**, 41 (2001).
[35] G. Milligan and D. Schilling, Multivariate Behav. Res. **20**, 97 (1985).
[36] D. Gfeller, J. C. Chappelier, and P. De Los Rios, Phys. Rev. E **72**, 056135 (2005).
[37] D. Wilkinson and B. Huberman, Proc. Natl. Acad. Sci. U.S.A. **101**, 5241 (2004).
[38] A. Arenas, L. Danon, A. Díaz-Guilera, P. Gleiser, and R. Guimerà, Eur. Phys. J. B **38**, 373 (2004).