Efficient and principled method for detecting communities in networks

Brian Ball,[1] Brian Karrer,[1] and M. E. J. Newman[1,2]
[1]*Department of Physics, University of Michigan, Ann Arbor, Michigan 48109, USA*
[2]*Center for the Study of Complex Systems, University of Michigan, Ann Arbor, Michigan 48109, USA*
(Received 27 April 2011; revised manuscript received 11 July 2011; published 8 September 2011)

A fundamental problem in the analysis of network data is the detection of network communities, groups of densely interconnected nodes, which may be overlapping or disjoint. Here we describe a method for finding overlapping communities based on a principled statistical approach using generative network models. We show how the method can be implemented using a fast, closed-form expectation-maximization algorithm that allows us to analyze networks of millions of nodes in reasonable running times. We test the method both on real-world networks and on synthetic benchmarks and find that it gives results competitive with previous methods. We also show that the same approach can be used to extract nonoverlapping community divisions via a relaxation method, and demonstrate that the algorithm is competitively fast and accurate for the nonoverlapping problem.

DOI: 10.1103/PhysRevE.84.036103 PACS number(s): 89.75.Hc, 02.10.Ox, 02.50.−r

I. INTRODUCTION

Many networked systems, including biological and social networks, are found to divide naturally into modules or communities, groups of vertices with relatively dense connections within groups but sparser connections between them [1,2]. Depending on context, the groups may be disjoint or overlapping. A fundamental problem in the theory of networks, and one that has attracted substantial interest among researchers in the last decade, is how to detect such communities in empirical network data [2,3]. There are a number of desirable properties that a good community detection scheme should have. First, it should be effective, meaning it should be able to accurately detect community structure when it is present. There are, for instance, many examples of networks, both naturally occurring and synthetic, for which the community structure is widely agreed upon, and a successful detection method should be able to find the accepted structure in such cases. Second, methods based on sound theoretical principles are preferable over those that are not. A method based on a mere hunch that something might work is inherently less trustworthy than one based on a provable result or fundamental mathematical insight. Third, when implemented as a computer algorithm, a method should ideally be fast and scale well with the size of the network analyzed. Many of the networks studied by current science are large, with millions or even billions of vertices, so a community detection algorithm whose running time scales, say, linearly with the size of the network is enormously preferred over one that scales as size squared or cubed.

In this paper we derive and demonstrate an algorithm for community detection in undirected, unweighted networks that can find either overlapping or nonoverlapping communities and satisfies all of the demands above. On standard benchmark tests the algorithm has performance similar to the best previous algorithms in detecting known community structure. The algorithm is based on established methods of statistical inference, namely, maximum likelihood and the expectation-maximization (EM) algorithm. And the algorithm is fast. In its simplest form it consists of the iteration of just two sets of equations, each iteration taking an amount of time that increases only linearly with system size. In practice the algorithm can handle networks with millions of vertices and edges in reasonable running times on a typical desktop computer: for the largest network we have analyzed, which has over 4 million vertices and 40 million edges, a single run of the algorithm takes less than an hour.

We approach the problem of community detection first as a problem of finding overlapping communities. Early efforts at community detection, going back to the 1970s, assumed nonoverlapping or disjoint communities [1,4,5], but as many researchers have argued in the last few years, it is common in practical situations for communities to overlap [6]. In social networks, for example, people often belong to more than one circle of acquaintances, e.g., family, friends, co-workers, and so forth, and hence those circles should properly be considered as overlapping since they have at least one common member. In biological networks too vertices can belong to more than one group. Metabolites in a metabolic network can play a role in more than one metabolic process or cycle; species in a food web can fall on the border between two otherwise noninteracting subcommunities and play a role in both of them. Thus the most general formulation of the community detection problem should allow for the possibility of overlap. Our approach is to develop a solution to this general problem first, then show how a variant of the same approach can be applied to nonoverlapping communities as well.

We tackle the detection of overlapping communities by fitting a stochastic generative model of network structure to observed network data. This approach, which applies methods of statistical inference to networks, has been explored by a number of authors for the nonoverlapping case, including some work that goes back several decades [5,7–9]. Extending the same approach to the overlapping case, however, has proved nontrivial. The crucial step is to devise a generative model that produces networks with overlapping community structure similar to that seen in real networks. The models used in most previous work are "mixed membership" models [10], in which, typically, vertices can belong to multiple groups and two vertices are more likely to be connected if they have more than one group in common. This, however, implies that the area of overlap between two communities should have a higher average density of edges than an area that falls in just a single community. It is unclear whether this reflects the

 ©2011 American Physical Society

behavior of real-world networks accurately, but it is certainly possible to construct networks that do not have this type of structure. Ideally we would prefer a less restrictive model that makes fewer assumptions about the structure of community overlaps.

Another set of approaches to the detection of overlapping communities are those based on local community structure [11]. Rather than splitting an entire network into communities in one step, these methods instead look for local groups within the network, based on analysis of local connection patterns. Methods of this kind give rise naturally to overlapping communities when one generates a large number of independent local communities throughout the network. Moreover, the communities tend to be compact and connected subgraphs, a requirement not always met by other methods. On the other hand, global detection methods can capture large-scale network structure better and are more appropriate when particular constraints, such as constraints on the number of communities, must be satisfied.

In this paper we develop a global statistical method for detecting overlapping communities based on the idea of link communities, which has been proposed independently by a number of authors both in the physics literature [12,13] and in machine learning [14,15]. The idea is that communities arise when there are different types of edges in a network. In a social network, for instance, there are links representing family ties, friendship, professional relationships, and so forth. If we can identify the types of the edges, i.e., if we can group not the vertices in a network but the edges, then we can deduce the communities of vertices after the fact from the types of edges connected to them. This approach has the nice feature of matching our intuitive idea of the origin and nature of community structure while giving rise to overlapping communities in a natural way: a vertex belongs to more than one community if it has more than one type of edge.

Previous approaches to the discovery of link communities have made use of heuristic quality functions optimized over possible partitions of a network's edges [12,13]. Such quality functions, particularly the so-called modularity function [16], have been used in the past for nonoverlapping communities, but while in practice these functions often give reasonable results, they also have some deficiencies: the modularity, for instance, can not be used to find very small communities [17], may not have a unique optimum [18], and is somewhat unsatisfactory from a formal viewpoint [19,20]. Recent results of Bickel and Chen [20] suggest that these deficiencies can be remedied by abandoning the quality function approach and instead fitting a generative model to the data. This is the approach we take, but the definition of a model for link communities entails some subtlety. In generative models for *vertex* communities, such as the mixed membership models mentioned above, one can assign vertices to groups first and then place edges based on that assignment. But for a model of link communities, where it is the edges that are partitioned, one can not assign edges to groups until the edges exist, so the edges and their groupings have to be generated simultaneously. We describe in detail how we achieve this in the following section. Once we have the model, the goal will be to determine the values of its parameters that best fit the observed network and from those to determine the overlapping vertex communities.

The outline of the paper is as follows. First, we define our model and then demonstrate how the best-fit values of its parameters can be calculated using a maximum likelihood algorithm. We also discuss how the algorithm can be implemented to optimize speed and memory requirements, allowing applications to large networks. We give example applications to numerous real-world networks, as well as tests against synthetic networks that demonstrate that the algorithm can discover known overlapping community structure in such networks.

Finally, we show how our method can be used also to detect nonoverlapping communities by assigning each vertex solely to the community to which it most strongly belongs in the overlapping division. We demonstrate that this intuitive heuristic can be justified rigorously by regarding the link community model as a relaxation of a stochastic blockmodel for disjoint communities [21]. Algorithms have been proposed previously for fitting this blockmodel, but their running time was typically at least quadratic in the number of vertices, which limited their application to smaller networks. The algorithm proposed here is significantly faster and hence can be applied to the detection of disjoint communities in very large networks.

II. A GENERATIVE MODEL FOR LINK COMMUNITIES

Our first step is to define the generative network model that we will use. The model generates networks with a given number n of vertices and undirected edges divided among a given number K of communities. It is convenient to think of the edges as being colored with K different colors to represent the communities to which they belong. Then the model is parametrized by a set of parameters θ_{iz}, which represent the propensity of vertex i to have edges of color z. Specifically, $\theta_{iz}\theta_{jz}$ is the expected number of edges of color z that lie between vertices i and j, the exact number being Poisson distributed about this mean value. Note that this means the network is technically a multigraph: it can have more than one edge between a pair of vertices. Some real-world networks contain such multiedges: in network representations of the World Wide Web, for instance, a single web page can contain several hyperlinks to the same other page. Most networks, however, have single edges only, and in this sense the model is unrealistic. However, allowing multiedges makes the model enormously simpler to treat and in practice the number of multiedges tends to be small, so the error introduced is also small, typically vanishing as $1/n$ in the limit of large network size. Multiedges are also allowed in most other random graph models of networks, such as the widely studied configuration model [22,23], and are neglected there for the same reasons. Our model also allows self-edges, edges that connect to the same vertex at both ends, with expected number $\frac{1}{2}\theta_{iz}\theta_{iz}$, the extra factor of a half being convenient for consistency with later results. Again, the appearance of self-edges, while unrealistic in some cases, greatly simplifies the mathematical developments and introduces only a small error.

In the model defined here, the link communities arise implicitly as the network is generated, as discussed in the introduction, rather than being spelled out explicitly. Two vertices i, j that have large values of θ_{iz} and θ_{jz} for some value of z have a high probability of being connected by an

edge of color z, and hence groups of such vertices will tend to be connected by relatively dense webs of color-z edges, precisely the structure we expect to see in a network with link communities.

III. DETECTING OVERLAPPING COMMUNITIES

Given the model defined above, it is now straightforward to write down the probability with which any particular network is generated. Recalling that a sum of independent Poisson-distributed random variables is also a Poisson-distributed random variable, the expected total number of edges of all colors between two vertices i and j is simply $\sum_z \theta_{iz}\theta_{jz}$ (or $\frac{1}{2}\sum_z \theta_{iz}\theta_{iz}$ for self-edges), and the actual number is Poisson distributed with this mean. Thus the probability of generating a graph G with adjacency matrix elements A_{ij} is

$$P(G|\theta) = \prod_{i<j} \frac{\left(\sum_z \theta_{iz}\theta_{jz}\right)^{A_{ij}}}{A_{ij}!} \exp\Bigl(-\sum_z \theta_{iz}\theta_{jz}\Bigr) \times \prod_i \frac{\left(\frac{1}{2}\sum_z \theta_{iz}\theta_{iz}\right)^{A_{ii}/2}}{(A_{ii}/2)!} \exp\left(-\tfrac{1}{2}\sum_z \theta_{iz}\theta_{iz}\right). \quad (1)$$

(Recall that the adjacency matrix element A_{ij}, by convention, takes the value $A_{ij} = 1$ if there is an edge between distinct vertices i and j, but $A_{ii} = 2$ for a self-edge—hence the additional factors of $\frac{1}{2}$ in the second product.)

We fit the model to an observed network by maximizing this probability with respect to the parameters θ_{iz}, or equivalently (and more conveniently) maximizing its logarithm. Taking the log of Eq. (1), rearranging, and dropping additive and multiplicative constants (which have no effect on the position of the maximum), we derive the log likelihood

$$\log P(G|\theta) = \sum_{ij} A_{ij} \log\left(\sum_z \theta_{iz}\theta_{jz}\right) - \sum_{ijz} \theta_{iz}\theta_{jz}. \quad (2)$$

Direct maximization of this expression by differentiating leads to a set of nonlinear implicit equations for θ_{iz} that are hard to solve, even numerically. An easier approach is the following. We apply Jensen's inequality in the form [24]

$$\log\left(\sum_z x_z\right) \geqslant \sum_z q_z \log\frac{x_z}{q_z}, \quad (3)$$

where the x_z are any set of positive numbers and the q_z are any probabilities satisfying $\sum_z q_z = 1$. Note that the exact equality can always be achieved by making the particular choice $q_z = x_z/\sum_z x_z$. Applying Eq. (3) to Eq. (2) gives

$$\log P(G|\theta) \geqslant \sum_{ijz}\left[A_{ij}q_{ij}(z)\log\frac{\theta_{iz}\theta_{jz}}{q_{ij}(z)} - \theta_{iz}\theta_{jz}\right], \quad (4)$$

where the probabilities $q_{ij}(z)$ can be chosen in any way we please provided they satisfy $\sum_z q_{ij}(z) = 1$. Notice that the $q_{ij}(z)$ are only defined for vertex pairs i,j that are actually connected by an edge in the network (so that $A_{ij} = 1$), and hence there are only as many of them as there are observed edges.

Since, as noted, the exact equality in this expression can always be achieved by a suitable choice of $q_{ij}(z)$, it follows that the double maximization of the right-hand side of (4) with respect to both the $q_{ij}(z)$ and the θ_{iz} is equivalent to maximizing the original log likelihood [Eq. (2)] with respect to the θ_{iz} alone. It may appear that this does not make our optimization problem any simpler: we have succeeded only in turning a single optimization into a double one, which one might well imagine was a more difficult problem. Delightfully, however, it is not; the double optimization is actually very simple. Given the true optimal values of θ_{iz}, the optimal values of $q_{ij}(z)$ are given by

$$q_{ij}(z) = \frac{\theta_{iz}\theta_{jz}}{\sum_z \theta_{iz}\theta_{jz}}, \quad (5)$$

since these are the values that make our inequality an exact equality. But, given the optimal values of the $q_{ij}(z)$, the optimal θ_{iz} can be found by differentiating (4), which gives

$$\theta_{iz} = \frac{\sum_j A_{ij}q_{ij}(z)}{\sum_i \theta_{iz}}. \quad (6)$$

Summing this expression over i and rearranging gives us

$$\left(\sum_i \theta_{iz}\right)^2 = \sum_{ij} A_{ij}q_{ij}(z), \quad (7)$$

and combining with (6) again then gives

$$\theta_{iz} = \frac{\sum_j A_{ij}q_{ij}(z)}{\sqrt{\sum_{ij} A_{ij}q_{ij}(z)}}. \quad (8)$$

Maximizing the log likelihood is now simply a matter of simultaneously solving Eqs. (5) and (8), which can be done iteratively by choosing a random set of initial values and alternating back and forth between the two equations. This type of approach is known as an expectation-maximization or EM algorithm and it can be proved that the log likelihood increases monotonically under the iteration, although it does not necessarily converge to the global maximum. To guard against the possibility of getting stuck at a local maximum, we repeat the entire calculation a number of times with random initial conditions and choose the result that gives the highest final log likelihood. In the work presented here, we found good results with numbers of repetitions in the range from 10 to 100.

The value of $q_{ij}(z)$ in Eq. (5) has a simple physical interpretation: it is the probability that an edge between i and j has color z, which is precisely the quantity we need in order to infer link communities in the network. Notice that $q_{ij}(z)$ is symmetric in i,j, as it should be for an undirected network.

The calculation presented here is mathematically closely related to methods developed in the machine learning community for the analysis of text documents. Specifically, the model we fit can be regarded as a variant of a model used in probabilistic latent semantic analysis (PLSA), a technique for automated detection of topics in a corpus of text, adapted to the present context of link communities. Connections between text analysis and community detection have been explored by several previous authors. Of particular interest is the work of Psorakis *et al.* [25], which, although it does not focus on link communities, uses another variant of the PLSA model, coupling it with an iterative fitting algorithm called nonnegative matrix factorization to find overlapping

communities in directed networks. Also of note is the work of Parkinnen *et al.* [14], who consider link communities as we do, but take a contrasting algorithmic approach based on a Bayesian generative model and Markov chain Monte Carlo techniques. A detailed description of the interesting connections between text processing and network analysis would take us some way from the primary purpose of this paper, but for the interested reader, we give a discussion and references in Appendix A.

IV. IMPLEMENTATION

The method outlined above can be implemented directly as a computer algorithm for finding overlapping communities, and works well for networks of moderate size, up to tens of thousands of vertices. For larger networks both memory usage and run time become substantial and prevent the application of the method to the largest systems, but both can be improved by using a more sophisticated implementation, which makes applications to networks of millions of vertices possible.

The algorithm's memory use is determined by the space required to store the parameters: the θ_{iz} require $O(nK)$ space while the $q_{ij}(z)$ require $O(mK)$, where n and m are the numbers of vertices and edges in the network. Since m is usually substantially larger than n, this means that memory use is dominated by the $q_{ij}(z)$. We can reduce memory use by reorganizing the algorithm in such a way that the $q_{ij}(z)$ are never stored. Rather than focusing on the θ_{iz}, we work instead with the average number k_{iz} of ends of edges of color z connected to vertex i:

$$k_{iz} = \sum_j A_{ij} q_{ij}(z). \tag{9}$$

Given the values of these quantities on a given iteration of the algorithm, the calculation of the values at the next iteration is then as follows. First, we define a new set of quantities k'_{iz} that will store the new values of the k_{iz}. Initially we set all of them to zero. We also calculate the average number κ_z of edges of color z summed over all vertices

$$\kappa_z = \sum_i k_{iz} \tag{10}$$

in terms of which the original θ_{iz} parameters are

$$\theta_{iz} = \frac{k_{iz}}{\sqrt{\kappa_z}}, \tag{11}$$

where we have used Eq. (8). Next we go through each edge (i,j) in the network in turn and calculate the denominator of Eq. (5) for that i and j from the values of the k_{iz}, thus:

$$D = \sum_z \theta_{iz}\theta_{jz} = \sum_z \frac{k_{iz}k_{jz}}{\kappa_z}. \tag{12}$$

Armed with this value, we can calculate the value of $q_{ij}(z)$ for this i,j and all z from Eq. (5):

$$q_{ij}(z) = \frac{\theta_{iz}\theta_{jz}}{\sum_z \theta_{iz}\theta_{jz}} = \frac{k_{iz}k_{jz}}{D\kappa_z}. \tag{13}$$

Now we add this value onto the quantities k'_{iz} and k'_{jz}, discard the values of D and $q_{ij}(z)$, and repeat for the next edge in the network. When we have gone through all edges in this manner, the quantities k'_{iz} will be equal to the sum in Eq. (9), and hence will be the correct new values of k_{iz}.

This method requires us to store only the old and new values of k_{iz}, for a total of $2nK$ quantities, and not the values of $q_{ij}(z)$. Depending on the values of m and n, this can result in substantial memory savings.

As for the running time, the algorithm as we have described it has a computational complexity of $O(mK)$ operations per iteration of the equations, where m is again the number of edges in the network, but this too can be improved. In a typical application of the algorithm to a network, the end result is that each vertex belongs to only a subset of the K possible communities. To put that another way, we expect that many of the parameters k_{iz} will tend to zero under the EM iteration. It is straightforward to see from the equations above that if a particular k_{iz} ever becomes zero, then it must remain so for all future iterations, which means that it no longer need be updated and we can save ourselves time by excluding it from our calculations. This leads to two useful strategies for pruning our set of variables. In the first, we set to zero any k_{iz} that falls below a predetermined threshold δ. Once a k_{iz} has been set to zero, the corresponding values of the $q_{ij}(z)$ on all the adjacent edges are also zero and therefore need not be calculated. Thus, for each edge, we need only calculate the values of $q_{ij}(z)$ for those colors z for which both k_{iz} and k_{jz} are nonzero, i.e., for the intersection of the sets of colors at vertices i and j. This strategy leads to speed increases when the number of communities $K \gtrsim 4$. For smaller values of K the speed savings are outweighed by the additional computational overhead and it is more efficient to simply calculate all $q_{ij}(z)$, but we nonetheless still set the values of the k_{iz} to zero below the threshold δ because it makes possible our second pruning strategy.

Our second strategy, which can be used in tandem with the first and gives significant speed improvements for all values of K, is motivated by the observation that if all but one of the k_{iz} for a particular vertex are set to zero, then the color of the vertex, meaning the group to which it belongs, is fixed at a single value and will no longer change at all. If both vertices at the ends of an edge (i,j) have this property, if both of them have converged to a single color and are no longer changing, then the edge connecting them no longer has any effect on the calculation and can be deleted entirely from the network.

By the use of these two strategies, the speed of our calculations is improved markedly. We find in practice that the numbers of parameters k_{iz} and edges both shrink rapidly and substantially with the progress of the calculation, so that the majority of the iterations involve only a subset, typically those associated with the vertices whose community identification is most ambiguous. If the value of the threshold δ is set to zero, then the pruned algorithm is exactly equivalent to the original EM algorithm and the results are identical, yet even with this choice we find substantial speed improvements. If δ is chosen small but nonzero (we use $\delta = 0.001$ in our calculations) then we introduce an approximation into the calculation, which means the results will be different in general from the original algorithm. In practice, however, the difference is small, and the nonzero δ gives us an additional and substantial speed

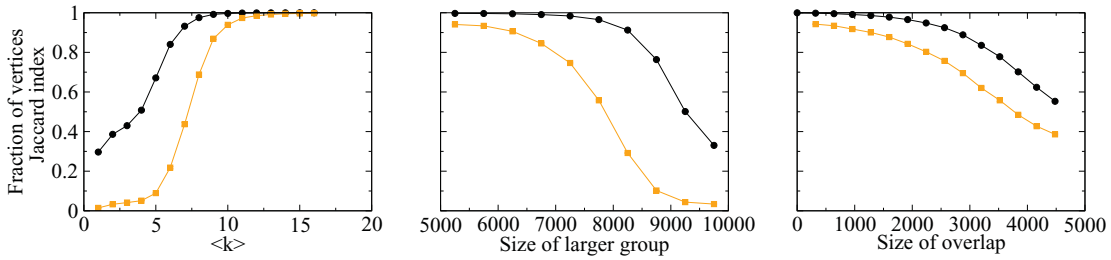


FIG. 1. (Color online) Results from the three sets of synthetic tests described in the text. Each data point is averaged over 100 networks. Twenty random initializations of the variables were used for each network and the run giving the highest value of the log likelihood was taken as the final result. In each panel the black curve shows the fraction of vertices assigned to the correct communities by the algorithm, while the lighter curve is the Jaccard index for the vertices in the overlap. Error bars are smaller than the points in all cases.

improvement. (In our experiments we find a variation of about 1% or less in the final log likelihood for values of δ anywhere from 0 to 0.1. Note, however, that if the value of δ is greater than $1/K$, then it is possible inadvertently to prune all of the colors from a vertex, leaving it in no community at all. To avoid this, one must choose $\delta < 1/K$.)

A detailed comparison of results and run times for the original and pruned versions of the algorithm is given in Appendix B for a range of networks. Unless stated otherwise, all calculations presented in the remainder of the paper are done with the faster version of the algorithm.

V. RESULTS

We have tested the performance of the algorithm described above using both synthetic (computer-generated) networks and a range of real-world examples. The synthetic networks allow us to test the algorithm's ability to detect known, planted community structure under controlled conditions, while the real networks allow us to observe performance under practical, real-world conditions.

A. Synthetic networks

Our synthetic network examples take the form of a classic consistency test. We generate networks using the same stochastic model that the algorithm itself is based on and measure the algorithm's ability to recover the known community divisions for various values of the parameters. One can vary the values to create networks with stark community structure (which should make detection easy) or no community structure at all (which makes it impossible), and everything in between, and we can thereby vary the difficulty of the challenge we pose to the algorithm.

The networks we use for our tests have $n = 10\,000$ vertices each, divided into two overlapping communities. We place x vertices in the first community only, meaning they have connections only to others in that community, y vertices in the second community only, and the remaining $z = n - x - y$ vertices in both communities, with equal numbers of connections to vertices in either group on average. We fix the expected degree of all vertices to take the same value k.

We perform three sets of tests. In the first, we fix the size of the overlap between the communities at $z = 500$, divide the remaining vertices evenly $x = y = 4750$, and observe the behavior of the algorithm as we vary the value of k. When $k \to 0$ there are no edges in the network and hence no community structure, and we expect the algorithm (or any algorithm) to fail. When k is large, on the other hand, it should be straightforward to work out where the communities are.

For our second set of tests, we again set the overlap at $z = 500$, but this time we fix $k = 10$ and vary the balance of vertices between x and y. Finally, for our third set of tests, we set $k = 10$ and constrain x and y to be equal, but allow the size z of the overlap to vary.

In Fig. 1 we show the measured fraction of vertices classified correctly (black curve) in each of these three sets of tests (the three separate panels), averaged over 100 networks for each point. To be considered correctly classified, a vertex's membership (or lack of membership) in both groups must be reported correctly by the algorithm, and the algorithm considers any vertex to be a member of a group if, on average, it has at least one edge of the appropriate color when the maximum likelihood fitting procedure is complete. In mathematical terms, a vertex belongs to community z if its expected degree with respect to color z, given by $\sum_j A_{ij} q_{ij}(z)$, is greater than one.

As the figure shows, there are substantial parameter ranges for all three tests for which the algorithm performs well, correctly classifying most of the vertices in the network. As expected, the accuracy in the first test increases with increasing k and for values of k greater than about 10, a figure easily attained by many real-world networks, the algorithm identifies the known community structure essentially perfectly. In the other two tests accuracy declines as either the asymmetry of the two groups or the size of the overlap increases, but approaches 100% when either is small.

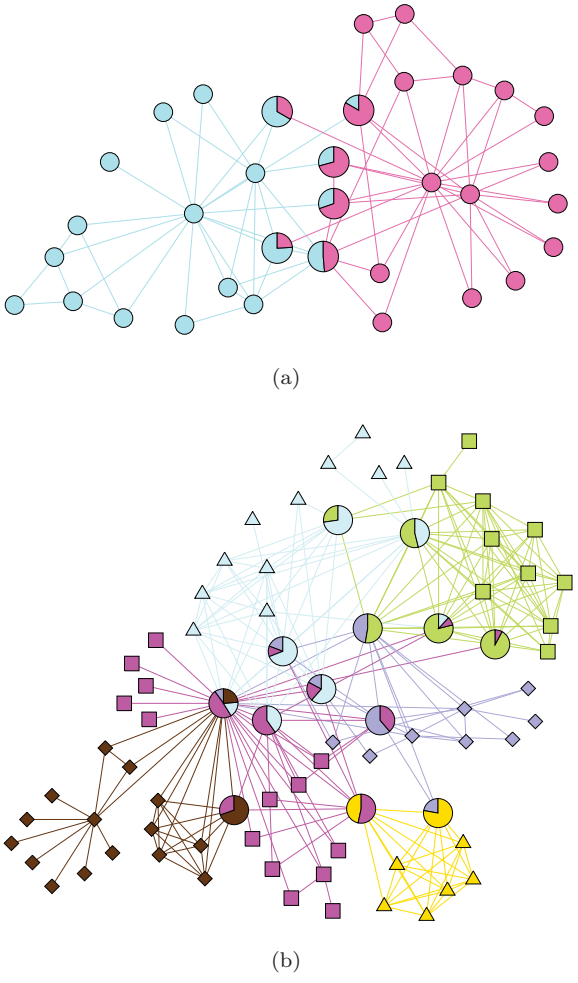
To probe in more detail the algorithm's ability to identify overlapping communities, we have also measured, for the same test networks, a Jaccard index: if S is the set of vertices in the true overlap and V is the set the algorithm identifies as being in the overlap, then the Jaccard index is $J = |S \cap V|/|S \cup V|$. This index is a standard measure of similarity between sets that rewards accurate identification of the overlap while penalizing both false positives and false negatives. The values of the index are shown as the lighter curves in Fig. 1 and, as we can see, the general shape of the curves is similar to the overall fraction of correctly identified vertices. In particular, we note that for networks with sufficiently high average degree k the value of J tends to 1, implying that the overlap is identified essentially perfectly.

B. Real networks

We have also tested our method on numerous real-world networks. In this section, we give detailed results for four specific examples. Summary results for a number of additional examples are given in Appendix B.

Our first example is one that has become virtually obligatory in tests of community detection, Zachary's "karate club" network, which represents friendship patterns between members of a university sports club, deduced from an observational study [26]. The network is interesting because the club split in two during the study, as a result of an internal dispute, and it has been found repeatedly that one can deduce the lines of the split from a knowledge of the network structure alone [1,2].

Figure 2(a) shows the decomposition of the karate club network into two overlapping groups as found by our algorithm. The colors in the figure show both the division of the vertices and the division of the edges. The split between the two groups in the club is clearly evident in the results and corresponds well with the acknowledged "ground truth," but in addition the algorithm assigns several vertices to both groups. The individuals represented by these overlap vertices, being by definition those who have friends in both camps, might be supposed to have had some difficulty deciding which side of the dispute to come down on, and indeed Zachary's original discussion of the split includes some indications that this was the case [26]. Note also that, in addition to identifying overlapping vertices, our method can assign to each a fraction by which it belongs to one community or the other, represented in the figure by the pie-chart coloring of the vertices in the overlap. The fraction is calculated as the expected fraction of edges of each color incident on the vertex.



(a)

(b)

FIG. 2. (Color online) Overlapping communities in (a) the karate club network of [26] and (b) the network of characters from *Les Misérables* [27], as calculated using the algorithm described in this paper. The edge colors correspond to the highest value of $q_{ij}(z)$ for the given edge, while vertex colors indicate the fraction of incident edges that fall in each community. Vertices in more than one community are drawn larger for clarity and divided into pie charts representing their division among communities.

Our second example is another social network and again one whose community structure has been studied previously. This network, compiled by Knuth [27], represents the patterns of interactions between the fictional characters in the novel *Les Misérables* by Victor Hugo. In this network two characters are connected by an edge if they appear in the same chapter of the book. Figure 2(b) shows our algorithm's partition of the network into six overlapping communities, and the partition accords roughly with social divisions and subplots in the plotline of the novel. But what is particularly interesting in this case is the role played by the hubs in the network, the major characters who are represented by vertices of especially high degree. It is common to find high-degree hubs in networks of many kinds, vertices with so many connections that they have links to every part of the network, and their presence causes problems for traditional, nonoverlapping community detection schemes because they do not fit comfortably in any community: no matter where we place a hub, it is going to have many connections to vertices in other communities. Overlapping communities provide an elegant solution to this problem because we can place the hubs in the overlaps. As Fig. 2(b) shows, our algorithm does exactly this, placing many of the hubs in the network in two or more communities. Such an assignment is in this case also realistic in terms of the plot of the novel: the major characters represented by the hubs are precisely those that appear in more than one of the book's subplots.

A similar behavior can be seen in our third example, which is a transportation network, the network of passenger airline flights between airports in the United States. In this network, based on data for flights in 2004, the vertices represent airports and an edge between airports indicates a regular scheduled direct flight. Spatial networks, those in which, as here, the vertices have well-defined positions in geographic space, are often found to have higher probability of connection for vertex pairs located closer together [28,29], which suggests that communities, if they exist, should be regional, consisting principally of blocks of nearby vertices. The communities detected by our algorithm in the airline network follow this pattern, as shown in Fig. 3. The three-way split shown divides the network into east and west coast groups and a group for Alaska. The overlaps are composed partly of vertices that lie along the geographic boundaries between the groups, but again include hubs as well, which tend to be placed in the overlaps even when they do not lie on boundaries. As with the previous example, this placement gives the algorithm a solution to the otherwise difficult problem of assigning to any

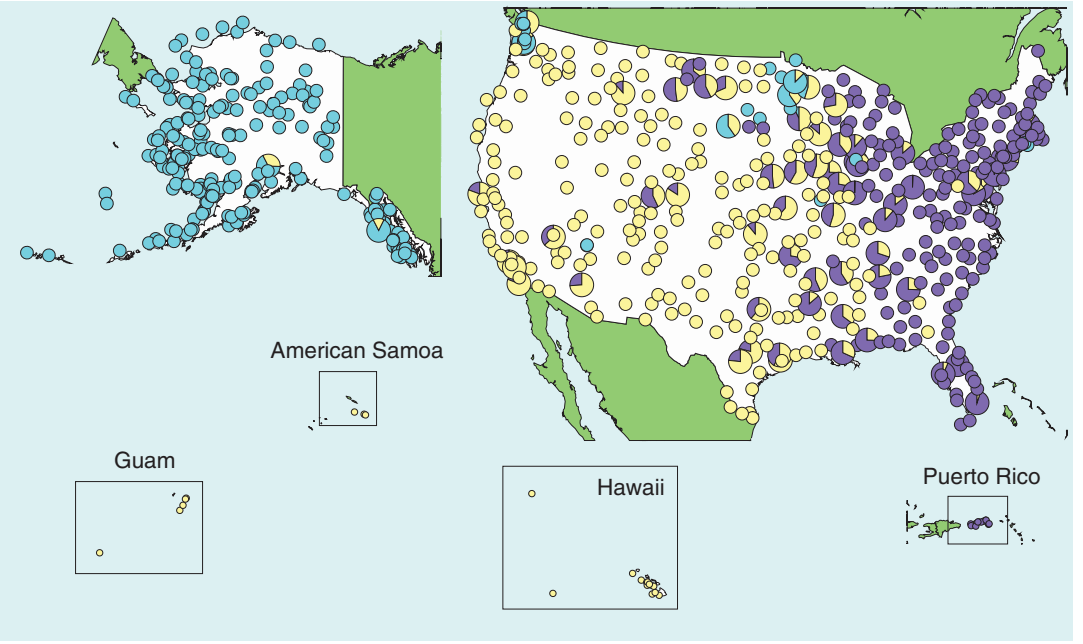


FIG. 3. (Color online) Overlapping communities in the network of US passenger air transportation. The three communities produced by the calculation correspond roughly to the east and west coasts of the country and Alaska.

one group a hub that has connections to all parts of the network. But, it also makes intuitive sense. Hubs are the "brokers" of the airline network, the vertices that connect different communities together, since they are precisely the airports that passengers pass through in traveling between distant locations. Thus it is appropriate that hubs be considered members of more than one group. In most cases the hubs belong most strongly to the community in which they are geographically located, and less strongly to other communities.

For our fourth example, we examine a network of coauthorships between researchers publishing on network science. In this network, which was previously published in [30], vertices represent scientists and unweighted edges connect pairs of scientists who have coauthored at least one paper together. Figure 4(a) shows the division of the network's largest component as found by our algorithm for $K = 12$ communities.

The figure reveals a new phenomenon not present in our previous examples: some of the communities found by the algorithm are not contiguous; they are divided into two or more separate parts with no edges connecting the parts. This seems unsatisfactory. Intuitively, one expects communities to be connected.

The explanation for this behavior is that in this case the algorithm has found a local optimum of the likelihood, rather than a global one, and the local optimum contains disconnected communities. To address this issue, we adopt the following procedure. After the communities are calculated with the EM algorithm we find all their connected clusters, then work through them in order from smallest to largest. Each cluster is added to the neighboring cluster (of any community) with which it has the most connections, unless it is the only cluster in its community, in which case we reverse the process and add the neighboring cluster to it. The only exception is when all neighboring clusters are the only cluster in their community, in which case we do nothing. Then we move on to the next largest cluster, bearing in mind that cluster sizes may have changed in the process. When we have gone through all clusters in this manner we are left with K communities, each of which is connected, consisting of a single cluster, and any connected pair of vertices that were originally assigned to the same cluster by the EM algorithm will still be in the same cluster.

This procedure requires very little additional effort to perform, and in our experiments we find that it always increases the likelihood of the community assignment, indicating that indeed the original EM algorithm found a local likelihood maximum. Figure 4(b) shows the result of applying the procedure to our coauthorship network and, as the figure shows, the communities found are now connected [31].

VI. NONOVERLAPPING COMMUNITIES

As we have described it, our algorithm is an algorithm for finding overlapping communities in networks, but it can be used to find nonoverlapping communities as well. As pointed out by a number of previous authors [25,32,33], any algorithm that calculates proportional membership of vertices in communities can be adapted to the nonoverlapping case by assigning each vertex to the single community to which it belongs most strongly. In our case, this means assigning vertices to the community for which the value of k_{iz}/κ_z is largest. It turns out that this procedure can be justified rigorously by regarding the link community model as a relaxation of a nonoverlapping degree-corrected stochastic blockmodel. The details are given in Appendix C. Here we give some example applications to show how the approach works in practice.

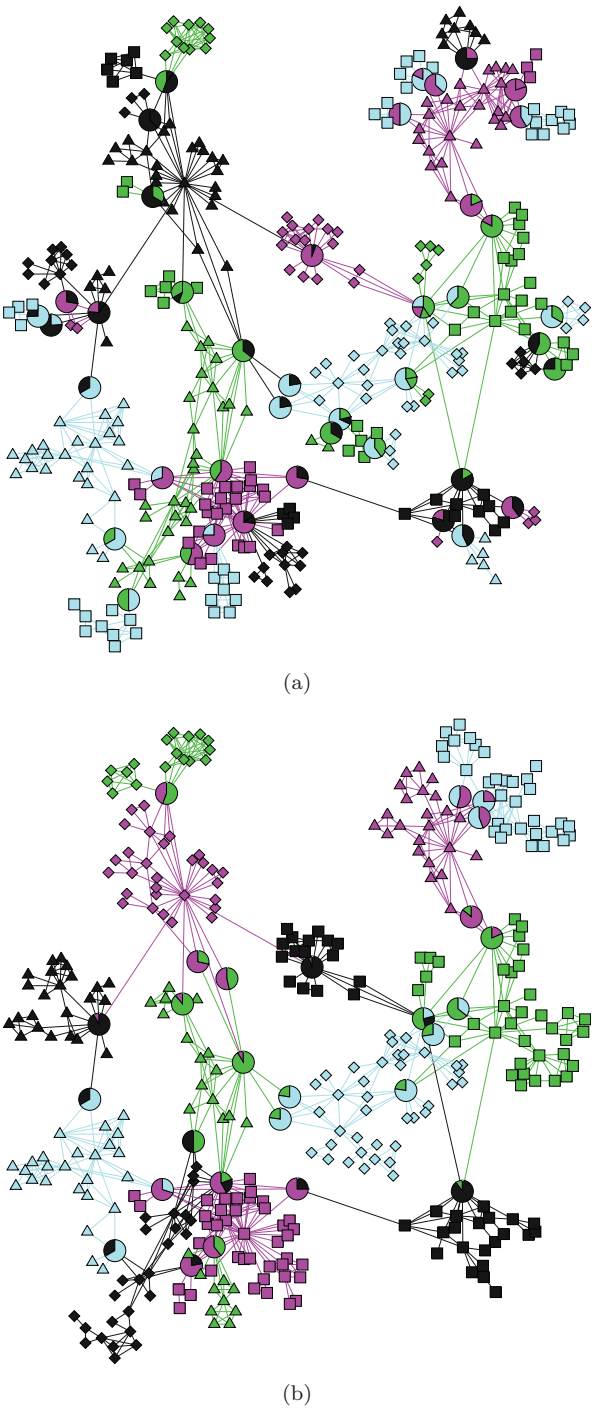


FIG. 4. (Color online) Overlapping communities in the collaboration network of network scientists as calculated by the algorithm of Sec. IV (a) without the post-processing step that ensures connected communities and (b) with the post processing. Each community is represented as a shape and color combination, except for overlapping vertices, which are always drawn as circles.

As with the overlapping case, we test the method on both synthetic and real-world networks. For the synthetic case, we use a standard test, the Lancichinetti-Fortunato-Radicchi or LFR benchmark for unweighted undirected networks with planted community structure [34,35]. To make possible comparisons with the previous study of Ref. [35] we use the same parameters, with networks of 1000 and 5000 vertices, average degree 20, maximum degree 50, degree exponent -2, and community exponent -1. We also use the same two ranges of community sizes, with communities of 10 to 50 vertices for one set of tests (labeled S for "small" in our figures) and 20 to 100 vertices for the other set (labeled B for "big"). The value of K for the detection algorithm was set equal to the number of communities in the benchmark network (which, because of the nature of the benchmark, is not a constant but varies from one network to another).

To quantify our algorithm's success at detecting the known communities in the benchmark networks we use the variant normalized mutual information measure proposed in [35]. We note that this measure is different, and in general returns different results, from the normalized mutual information measure most often used to evaluate community structure [3], but using it allows us to make direct comparisons with the results for other algorithms given in [35].

In our benchmark tests we find that the method described above for finding nonoverlapping communities, i.e., just choosing the community with the highest value of k_{iz}/κ_z, returns only average performance when compared with the other algorithms tested in Ref. [35]. However, a simple modification of the algorithm produces significantly better results: after generating a candidate division into communities using the rounding method, we then apply a further optimization step in which move each vertex to the community that gives the largest value of the log likelihood of the division under the stochastic blockmodel, and repeat this exercise until no further such moves exist. This process, which is reminiscent of the well-known Kernighan-Lin algorithm for graph partitioning [36], is easy to implement and carries little computational cost when compared to the calculation of the initial division, but it improves our results dramatically.

The results of our tests are shown in Fig. 5. The top panel shows the performance of the algorithm without the additional optimization step and the results fall in the middle of the pack when compared to previous algorithms, better than some methods but not as good as others. The bottom panel shows the results with the additional optimization step added, and now the algorithm performs about as well as, or better than, the algorithms analyzed in Ref. [35]. The general shape of the mutual information curve is similar to that of the best competing methods, falling off around the same place, although the mutual information values are somewhat lower for low values of the mixing parameter, indicating that the method is not getting the community structure exactly correct in this regime. Examining the communities in detail reveals that the method occasionally splits or merges communities. It is possible that performance could be improved further by a less simple-minded post-processing step for optimizing the likelihood. In particular, by contrast with the overlapping groups of the preceding section, we made no effort to ensure that the communities in the present tests consisted of only a single cluster, and doing so might potentially improve the results.

We also give, in Fig. 6, an example of a test of the method against a real-world network, in this case the much studied college football network of Ref. [1]. In this network the vertices represent university teams in American football and the edges represent the schedule of games for the year 2000 football

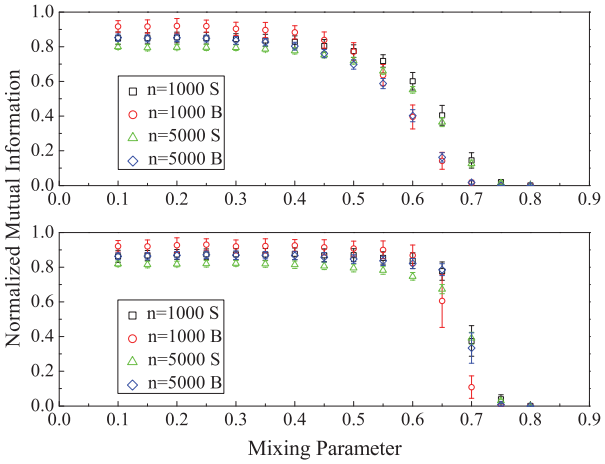


FIG. 5. (Color online) Performance of the nonoverlapping community algorithm described in the text when applied to synthetic networks generated using the LFR benchmark model of Lancichinetti *et al.* [35]. Parameters used are the same as in Ref. [35] and (S) and (B) denote networks with the "small" and "big" community sizes used by the same authors. The top and bottom panels, respectively, show the results without and with post processing to optimize the value of the log likelihood. Ten random initializations of the variables were used for each network and each point is an average over 100 networks.

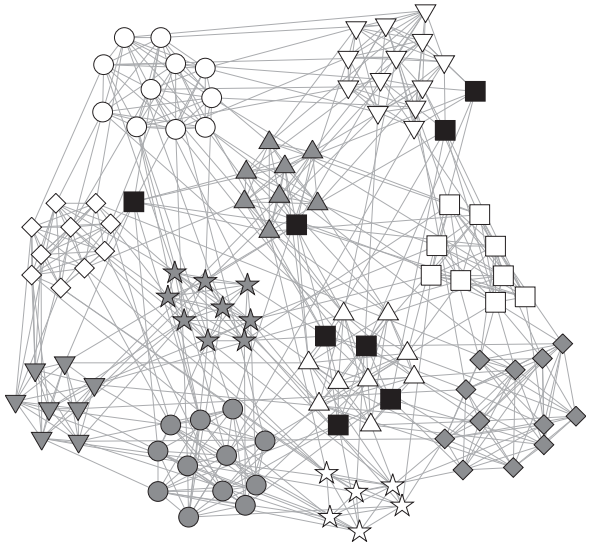


FIG. 6. Nonoverlapping communities found in the US college football network of Ref. [1]. The clusters of vertices represent the communities found by the algorithm, while the vertex shape and color combination represents the "conferences" into which the colleges are formally divided. As we can see, the algorithm in this case extracts the known conference structure perfectly. (The square black vertices represent independent colleges that belong to no conference.)

season, two teams being connected if they played a game. It has been found in repeated analyses that a clustering of this network into communities can retrieve the organizational units of US college sports, called "conferences," into which universities are divided for the purposes of competition. In 2000 there were 11 conferences among the Division I-A teams that make up the network, as well as 8 teams independent of any conference. As Fig. 6 shows, every single team that belongs in a conference is placed correctly by our algorithm.

VII. CONCLUSION

In this paper we have described a method for detecting communities, either overlapping or not, in undirected networks. The method has a rigorous mathematical foundation, being based on a probabilistic model of link communities, is easy to implement, fast enough for networks of millions of vertices, and gives results competitive with other algorithms.

Nonetheless, the method is not perfect. Its main current drawback is that it offers no criterion for determining the value of the parameter we call K, the number of communities in a network. This is a perennial problem for community detection methods of all kinds. Some methods, such as modularity maximization, do offer a solution to the problem, but that solution is known to give biased answers or be inconsistent under at least some circumstances [17,20]. More rigorous approaches, such as the Bayesian information criterion [37] and the Akaike information criterion [38], are unfortunately not applicable here because many of the model parameters are zero, putting them on the boundary of the parameter space, which invalidates the assumptions made in deriving these criteria.

Another approach to choosing the value of K is to perform the calculations with a large value and regularize the parameters in a manner such that some communities disappear, meaning that zero edges are associated with those communities. For example, Psorakis *et al.* [25], in studies using their matrix factorization algorithm, used priors that penalized their model for including too many nonzero parameter values and hence created a balance between numbers of communities and goodness of fit to the network data. Unfortunately, the priors themselves contain undetermined parameters whose values can influence the number of communities and hence the problem is not completely solved by this approach.

We believe that statistical model selection methods applied to generative models should in principle be able to find the number of communities in a consistent and satisfactory manner. We have performed some initial experiments with such methods and the quality of the results seems promising, but the methods are at present too computationally demanding to be applied to any but the smallest of networks. It is an open question whether a reliable method can be developed that runs in reasonable time on the large networks of interest to today's scientists.

ACKNOWLEDGMENTS

The authors thank Q. Mei, C. Moore, and L. Zdeborova for useful conversations. This work was funded in part by the National Science Foundation under Grant No. DMS–0804778 and by the James S. McDonnell Foundation.

APPENDIX A: COMMUNITY DETECTION AND STATISTICAL TEXT ANALYSIS

As mentioned in the main text, the generative model we use is the network equivalent of a model used in the text analysis technique called probabilistic latent semantic analysis (PLSA)

[39–41], modified somewhat for the particular problem we are addressing. In this appendix we describe PLSA and related methods and models and their relationship to the community detection problem.

A classic problem in text analysis, which is addressed by the PLSA method, is that of analyzing a "corpus" of text documents to find sets of words that all (or mostly) occur in the same documents. The assumption is that these sets of words correspond to topics or themes that can be used to group documents according to content. The PLSA approach regards documents as a so-called "bag of words," meaning one considers only the number of times each word occurs in a document and not the order in which words occur. (Also, one often considers only a subset of words of interest, rather than all words that appear in the corpus.)

Mathematically, a corpus of D documents and W words of interest is represented by a matrix A having elements A_{wd} equal to the number of times word w appears in document d. To make the connection to networks, this matrix can be thought of as the incidence matrix of a weighted bipartite network having one set of vertices for the documents, another for the words, and edges connecting words to the documents in which they appear with weight equal to their frequency of occurrence.

In PLSA each word-document pair (an edge in the corresponding network picture) is associated with an unobserved variable z which denotes one of K topical groups. Each edge is assumed to be placed independently at random in the bipartite graph, with the probability that an edge falls between word w and document d being broken down in the form $\sum_z P(w|z)P(d|z)P(z)$, where $P(z)$ is the probability that the edge belongs to topic z, $P(w|z)$ is the probability that an edge with topic z connects to word w, and $P(d|z)$ is the probability that an edge with topic z connects to document d. Note that, given the topic, the document and word ends of each edge are placed independently. (Hofmann [39] calls this parametrization a "symmetric" one, meaning that the word and the document play equivalent roles mathematically, but in the networks jargon, this would not be considered a symmetric formulation: the network is bipartite and the incidence matrix is not symmetric, nor even, in general, square.)

An alternative description of the model, which is useful for actually generating the incidence matrix and which corresponds with our formulation of the equivalent network problem, is that each matrix element A_{wd} takes a random value drawn independently from a Poisson distribution with mean $\sum_z P(w|z)P(d|z)\,\omega_z$. In the language of networks, each edge is placed with independent probability $\sum_z P(w|z)P(d|z)P(z)$, where $P(z) = \omega_z / \sum_{z'} \omega_{z'}$. In our work, where we focus on one-mode networks and a symmetric adjacency matrix instead of an incidence matrix, the parameter ω_z is redundant and we omit it.

PLSA involves using the edge probability above to calculate a likelihood for the entire word-document distribution, then maximizing with respect to the unknown probabilities $P(w|z)$, $P(d|z)$, and $P(z)$. The resulting probabilities give one a measure of how strongly each word or document is associated with a particular topic z, but since the topics are arbitrary, this is effectively the same as simply grouping the words and documents into "communities." Alternatively, one can use the probabilities to divide the edges of the bipartite graph among the topical groups, giving the text equivalent of the "link communities" that are the focus of our calculations.

A number of methods have been explored for maximizing the likelihood. Mathematically the one most closely related to our approach is the expectation-maximization (EM) algorithm of Hofmann [39–41], although the correspondence is not exact. Hofmann's work focuses solely on text processing—the connection to networks was not made until later—and because of its inherently asymmetric form the method can not be translated directly for applications to standard one-mode networks. Instead we must reformulate the problem using a symmetric model, which leads to the approach described in this paper. The symmetric formulation and the corresponding EM algorithm have not, to our knowledge, been used previously for community detection in networks, but several other related approaches have, including ones based on the techniques known as non-negative matrix factorization (NMF) [42,43] and latent Dirichlet allocation (LDA) [44,45]. These formulations have similar goals to ours, but are typically asymmetric (and hence unsuitable for undirected networks) and use different algorithmic approaches for maximizing the likelihood. The NMF formulation is similar in style to an EM algorithm, using an iterative maximization scheme, but the specific iteration equations are different. Several papers have recently proposed using NMF to find overlapping communities [25,32,33], and in particular the work of Psorakis *et al.* [25] mentioned in the main text uses NMF with the PLSA model, although again in an asymmetric formulation, and not applied to link communities.

Recent work by Parkinnen *et al.* [14] and Gyenge *et al.* [15] does consider link communities, in an asymmetric formulation, but uses algorithmic approaches that are different again. For instance, Parkinnen *et al.* [14] use a model that attaches conjugate priors to the parameters and then samples the posterior distribution of link communities with a collapsed Gibbs sampler.

LDA [44,45] offers an alternative but related approach that also attaches priors to the parameters, but in a specific way that relies on the asymmetric formulation of the model. In [46] and [47] LDA is adapted to networks by treating vertex-edge pairs as analogous to word-document pairs and then associating communities with the vertex-edge pairs. This is an interesting approach but differs substantially from the others discussed here, including our own, in which vertex-vertex pairs (i.e., edges) are the quantity analogous to word-document pairs.

Finally, in Appendix C we show that our model can be used to find nonoverlapping communities by viewing it as a relaxation of a nonoverlapping stochastic blockmodel. A corresponding relaxation has been noted previously for a version of NMF and was shown to be related to spectral clustering [48,49].

APPENDIX B: RESULTS FOR RUNNING TIME

As discussed in Sec. IV, a naive implementation of the EM equations gives an algorithm that is only moderately fast—not fast enough for very large networks. We described a more sophisticated implementation that prunes unneeded variables from the iteration and achieves significantly greater speed. In this appendix, we give a comparison of the performance of the two versions of the algorithm on a set of test networks.

TABLE I. Example networks and running times for each of the three versions of the overlapping communities algorithm described in the text. The designations "fast" and "naive" refer to the algorithm with and without pruning, respectively. "Iterations" refer to the total number of iterations for the entire run, not the average number for one random initialization. "Time" is similarly the total running time for all initializations. Directed networks were symmetrized for these tests. All networks were run with 100 random initializations, except for the LiveJournal network, which was run with 10 random initializations. Calculations were run on one core of a four-core 3.2-GHz Intel Core i5 CPU with 4 GB memory under the Red Hat Enterprise Linux operating system. Running times do not include the additional cluster aggregation process described in Sec. V B, but in practice the extra time for this process is negligible.

Running conditions	Time (s)	Iterations	Log likelihood
US air transportation, $n = 709$, $m = 3327$, $K = 3$			
Naive, $\delta = 0$	15.71	55 719	−8924.58
Fast, $\delta = 0$	14.67	55 719	−8924.58
Fast, $\delta = 0.001$	2.17	26 063	−9074.21
Network science collaborations [30], $n = 379$, $m = 914$, $K = 3$			
Naive, $\delta = 0$	0.93	13 165	−3564.74
Fast, $\delta = 0$	0.82	13 165	−3564.74
Fast, $\delta = 0.001$	0.13	10 747	−3577.85
Network science collaborations, $n = 379$, $m = 914$, $K = 10$			
Naive, $\delta = 0$	3.19	18 246	−2602.15
Fast, $\delta = 0$	3.15	18 246	−2602.15
Fast, $\delta = 0.001$	0.49	12 933	−2611.96
Network science collaborations, $n = 379$, $m = 914$, $K = 20$			
Naive, $\delta = 0$	6.16	19 821	−2046.95
Fast, $\delta = 0$	6.09	19 821	−2046.95
Fast, $\delta = 0.001$	0.94	14 010	−2094.85
Political blogs [50], $n = 1490$, $m = 16\,778$, $K = 2$			
Naive, $\delta = 0$	11.42	13 773	−48761.1
Fast, $\delta = 0$	11.46	13 773	−48761.1
Fast, $\delta = 0.001$	4.14	13 861	−48765.6
Physics collaborations [51], $n = 40\,421$, $m = 175\,693$, $K = 2$			
Naive, $\delta = 0$	4339.57	424 077	-1.367×10^6
Fast, $\delta = 0$	2557.91	424 077	-1.367×10^6
Fast, $\delta = 0.001$	253.41	61 665	-1.378×10^6
Amazon copurchasing [52], $n = 403\,394$, $m = 2\,443\,408$, $K = 2$			
Naive, $\delta = 0$	170 646.9	1 222 937	-2.521×10^7
Fast, $\delta = 0$	105 042.3	1 222 937	-2.521×10^7
Fast, $\delta = 0.001$	11 635.0	120 612	-2.538×10^7
LiveJournal [53,54], $n = 4\,847\,571$, $m = 42\,851\,237$, $K = 2$			
Fast, $\delta = 0$	333 230	278 707	-4.611×10^8
Fast, $\delta = 0.001$	33 924	19 257	-4.642×10^8

The results are summarized in Table I, which gives the CPU time in seconds taken to complete the overlapping community detection calculation on a standard desktop computer for each of the test networks. In these tests we use 100 random initializations of the variables and take as our final result the run that gives the highest value of the log likelihood. For each network we give the results of three different calculations: (1) the calculation performed using the naive EM algorithm; (2) the calculation using the pruned algorithm with the threshold parameter δ set to zero, meaning the algorithm gives results identical to the naive algorithm except for numerical rounding, but runs faster; and (3) the calculation performed using the pruned algorithm with $\delta = 0.001$, which introduces an additional approximation that typically results in a slightly poorer final value of the log likelihood, but gives a significant additional boost in speed.

The largest network studied, which is a network of links in the online community LiveJournal, is an exception to the pattern: for this network, which contains over 40 million edges, we performed runs with only 10 random initializations each, using the pruned algorithm with $\delta = 0.001$ and with $\delta = 0$. Each randomly initialized run took about 50 min to complete for $\delta = 0.001$ and about 9 h for $\delta = 0$.

While the algorithm described is fast by comparison with most other community detection methods, it is possible that its speed could be improved further (or that the quality of the results could be improved while keeping the speed the same). Two potential improvements are suggested by the text processing literature discussed in Appendix A. The first, from Hofmann [41], is to use the so-called tempered EM algorithm. The second, from Ding *et al.* [43], is to alternate between the EM algorithm and a non-negative matrix factorization algorithm, exploiting the fact that both maximize the same objective function but in different ways.

APPENDIX C: NONOVERLAPPING COMMUNITIES

In Sec. VI we described a procedure for extracting nonoverlapping community assignments from network data by first finding overlapping ones and then assigning each vertex to the community to which it belongs most strongly. This procedure was presented as a heuristic strategy for the nonoverlapping problem, but in this appendix we show that it can be derived in a principled manner as an approximation method for fitting the data to a degree-corrected stochastic blockmodel.

Methods have been proposed for discovering nonoverlapping communities in networks by fitting to the class of models known as stochastic blockmodels. As discussed in Ref. [21], it turns out to be crucial that the blockmodel used incorporate knowledge of the degree sequence of the network if it is to produce useful results, and this leads us to consider the so-called degree-corrected blockmodel, which can be formulated as follows. We consider a network of n vertices, with each vertex belonging to exactly one community. The community assignments are represented by an indicator variable S_{ir} which takes the value 1 if vertex i belongs to community r and zero otherwise. To generate the network, we place a Poisson-distributed number of edges between each pair of vertices i, j, such that the expected value of the adjacency matrix element A_{ij} is $\theta_i \omega_{rs} \theta_j$ if vertex i belongs to group r and vertex j belongs to group s, where θ_i and ω_{rs} are parameters of the model. To put this another way, the expected value of the adjacency matrix element is $\theta_i(\sum_{rs} S_{ir}\omega_{rs}S_{js})\theta_j$ for every vertex pair. The normalization of the parameters is arbitrary since we can rescale all θ_i by the same constant if we simultaneously rescale all ω_{rs}. In our calculations we fix the normalization so that the θ_i sum to unity within each community: $\sum_i \theta_i S_{ir} = 1$ for all r.

Now one can fit this model to an observed network by writing the probability of generation of the network as a product of Poisson probabilities for each (multi)edge, then maximizing with respect to the parameters θ_i and ω_{rs} and the community assignments S_{ir}. Unfortunately, while the maximization with respect to the continuous parameters θ_i and ω_{rs} is a simple matter of differentiation, the maximization with respect to the discrete variables S_{ir} is much harder. A common way around such problems is to "relax" the discrete variables, allowing them to take on continuous real values, so that the optimization can be performed by differentiation. In the present case, we allow the S_{ir} to take on arbitrary non-negative values, subject to the constraint that $\sum_r S_{ir} = 1$. In effect, S_{ir} now represents the fraction by which vertex i belongs to group r, with the constraint ensuring that the fractions add correctly to 1.

With this relaxation, we can now absorb the parameters θ_i into the S_{ir}, defining $\theta_{ir} = \theta_i S_{ir}$ with $\sum_i \theta_{ir} = 1$, and the mean number of edges between vertices i and j becomes $\sum_{rs} \theta_{ir}\omega_{rs}\theta_{js}$. This is an extended form of the overlapping communities model studied in this paper, generalized to include the extra $K \times K$ matrix ω_{rs}. In the language of link communities, this generalization gives us a model in which the two ends of an edge can belong to different communities. One can think of each end of the edge as being colored with its own color, instead of the whole edge taking only a single color. If ω_{rs} is constrained to be diagonal then we recover the single-color version of the model again.

We can fit the general (nondiagonal) model to an observed network using an expectation-maximization algorithm, just as before. Defining a probability $q_{ij}(r,s)$ that an edge between i and j has colors r and s, the EM equations are now

$$q_{ij}(r,s) = \frac{\theta_{ir}\omega_{rs}\theta_{js}}{\sum_{rs} \theta_{ir}\omega_{rs}\theta_{js}} \tag{C1}$$

and

$$\theta_{ir} = \frac{\sum_{js} A_{ij} q_{ij}(r,s)}{\sum_{ijs} A_{ij} q_{ij}(r,s)}, \quad \omega_{rs} = \sum_{ij} A_{ij} q_{ij}(r,s). \tag{C2}$$

By iterating these equations we can find a solution for the parameters θ_{ir}. But $\theta_{ir} = \theta_i S_{ir}$ and, summing both sides over r, we get $\sum_r \theta_{ir} = \theta_i$ since $\sum_r S_{ir} = 1$. Hence,

$$S_{ir} = \frac{\theta_{ir}}{\theta_i} = \frac{\theta_{ir}}{\sum_r \theta_{ir}}. \tag{C3}$$

Thus we can calculate the values of S_{ir} and once we have these we can then reverse the relaxation of the model by rounding the values to zero or one, which is equivalent to assigning each vertex i to the community r for which S_{ir} is largest, or equivalently the community for which θ_{ir} is largest.

Thus the final algorithm for dividing the network is simply to iterate the EM equations to convergence and then assign each vertex to the community for which θ_{ir} is largest. In the language of Sec. VI, this is equivalent to looking for the largest value of k_{iz}/κ_z, and hence this algorithm is the same as the algorithm that we described in that section, except that the model is generalized to include the matrix ω_{rs}, where in our original calculations this matrix was absent, which is equivalent to assuming it to be diagonal. In our experiments, however, we have found that even when we allow ω_{rs} to be nondiagonal, the algorithm usually chooses a diagonal value anyway, which implies that the output of our original algorithm and the generalized algorithm should be the same. (We note that in practice the diagonal version of the algorithm runs faster, while both are substantially faster than the vertex moving heuristic proposed for the stochastic blockmodel in Ref. [21].)

Diagonal values are expected for networks with traditional community structure, where connections are more dense within communities than between them. It is entirely possible, however, that there could be networks with interesting nondiagonal group structure that could be detected using the more general model. The model including the matrix ω_{rs} can in principle find disassortative community structure (structure in which connections are less common within communities than between them) as well as the better studied assortative structure. For example, it can detect bipartite structure in networks, whereas the unadjusted model can not.

[1] M. Girvan and M. E. J. Newman, Proc. Natl. Acad. Sci. USA **99**, 7821 (2002).

[2] S. Fortunato, Phys. Rep. **486**, 75 (2010).

[3] L. Danon, J. Duch, A. Diaz-Guilera, and A. Arenas, J. Stat. Mech.: Theory Exp. (2005) P09008.

[4] R. L. Breiger, S. A. Boorman, and P. Arabie, J. Math. Psychol. **12**, 328 (1975).

[5] P. W. Holland, K. B. Laskey, and S. Leinhardt, Social Networks **5**, 109 (1983).

[6] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Nature (London) **435**, 814 (2005).

[7] Y. J. Wang and G. Y. Wong, J. Am. Stat. Assoc. **82**, 8 (1987).

[8] T. A. Snijders and K. Nowicki, J. Class. **14**, 75 (1997).

[9] A. Clauset, C. Moore, and M. E. J. Newman, Nature (London) **453**, 98 (2008).

[10] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, J. Mach. Learn. Res. **9**, 1981 (2008).

[11] J. P. Bagrow, J. Stat. Mech.: Theory Exp. (2008) P05001.

[12] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, Nature (London) **466**, 761 (2010).

[13] T. S. Evans and R. Lambiotte, Phys. Rev. E **80**, 016105 (2009).

[14] J. Parkinnen, A. Gyenge, J. Sinkkoken, and S. Kaski, in *Proceedings of the 7th International Workshop on Mining and Learning with Graphs* (Association of Computing Machinery, New York, 2009).

[15] A. Gyenge, J. Sinkkonen, and A. A. Benczúr, in *Proceedings of the 8th International Workshop on Mining and Learning with Graphs* (Association of Computing Machinery, New York, 2010), pp. 62–69.

[16] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).

[17] S. Fortunato and M. Barthélémy, Proc. Natl. Acad. Sci. USA **104**, 36 (2007).

[18] B. H. Good, Y.-A. de Montjoye, and A. Clauset, Phys. Rev. E **81**, 046106 (2010).
[19] X. S. Zhang, R. S. Wang, Y. Wang, J. Wang, Y. Qiu, L. Wang, and L. Chen, Europhys. Lett. **87**, 38002 (2009).
[20] P. J. Bickel and A. Chen, Proc. Natl. Acad. Sci. USA **106**, 21068 (2009).
[21] B. Karrer and M. E. J. Newman, Phys. Rev. E **83**, 016107 (2011).
[22] M. Molloy and B. Reed, Random Struct. Algorithms **6**, 161 (1995).
[23] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, Phys. Rev. E **64**, 026118 (2001).
[24] This is a special case of the general observation that the log of the average of any set of numbers is never less than the average of the log since the logarithm function is concave down. If the numbers in question are x_z/q_z and the average is taken with weights q_z, this observation leads immediately to the inequality given.
[25] I. Psorakis, S. Roberts, M. Ebden, and B. Sheldon, Phys. Rev. E **83**, 066114 (2011).
[26] W. W. Zachary, J. Anthropol. Res. **33**, 452 (1977).
[27] D. E. Knuth, *The Stanford GraphBase: A Platform for Combinatorial Computing* (Addison–Wesley, Reading, MA, 1993).
[28] M. T. Gastner and M. E. J. Newman, Eur. Phys. J. B **49**, 247 (2006).
[29] M. Barthélémy, Phys. Rep. **499**, 1 (2011).
[30] M. E. J. Newman, Phys. Rev. E **74**, 036104 (2006).
[31] We have also applied the same procedure to the previous example networks, including the synthetically generated ones, but it produced in no significant changes to the results in those cases.
[32] M. Zarei, D. Izadi, and K. A. Samani, J. Stat. Mech.: Theory Exp. (2009) P11013.
[33] F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding, Data Mining Know. Disc. **22**, 493 (2011).
[34] A. Lancichinetti, S. Fortunato, and F. Radicchi, Phys. Rev. E **78**, 046110 (2008).
[35] A. Lancichinetti and S. Fortunato, Phys. Rev. E **80**, 056117 (2009).
[36] B. W. Kernighan and S. Lin, Bell Syst. Tech. J. **49**, 291 (1970).
[37] G. Schwarz, Ann. Stat. **6**, 461 (1978).
[38] H. Akaike, IEEE Trans. Autom. Control **19**, 716 (1974).
[39] T. Hofmann, in *Proceedings of the 22nd Annual International ACM Conference on Research and Development in Information Retrieval* (Association of Computing Machinery, New York, 1999), pp. 50–57.
[40] T. Hofmann, Mach. Learn. **42**, 177 (2001).
[41] T. Hofmann, ACM Trans. Inf. Syst. **22**, 89 (2004).
[42] D. D. Lee and H. S. Seung, Nature (London) **401**, 788 (1999).
[43] C. Ding, T. Li, and W. Peng, Comput. Stat. Data Anal. **52**, 3913 (2008).
[44] D. M. Blei, A. Y. Ng, and M. I. Jordan, J. Mach. Learn. Res. **3**, 993 (2003).
[45] M. Girolami and A. Kabán, in *Proceedings of the 26th Annual International ACM Conference on Research and Development in Information Retrieval* (Association of Computing Machinery, New York, 2003), pp. 433–434.
[46] K. Henderson and T. Eliassi-Rad, in *Proceedings of the 2009 ACM Symposium on Applied Computing* (Association of Computing Machinery, New York, 2009), pp. 1456–1461.
[47] H. Zhang, B. Qiu, C. L. Giles, H. C. Foley, and J. Yen, in *Proceedings of the IEEE International Conference on Intelligence and Security Informatics* (Institute of Electrical and Electronics Engineers, New York, NY, 2007), pp. 200–207.
[48] C. Ding, X. He, and H. D. Simon, in *Proceedings of the SIAM Data Mining Conference* (Society for Industrial and Applied Mathematics, Philadelphia, PA, 2005), pp. 606–610.
[49] C. Ding, T. Li, and M. Jordan, in *Proceedings of the 8th IEEE International Conference on Data Mining* (Institute of Electrical and Electronics Engineers, New York, NY, 2008), pp. 183–192.
[50] L. A. Adamic and N. Glance, in *Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem* (Association for Computing Machinery, New York, NY, 2005).
[51] M. E. J. Newman, Proc. Natl. Acad. Sci. USA **98**, 404 (2001).
[52] J. Leskovec, L. A. Adamic, and B. A. Huberman, ACM Trans. Web **1**, 5 (2007).
[53] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, in *Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining* (Association of Computing Machinery, New York, 2006), pp. 44–54.
[54] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, e-print arXiv:0810.1355.