Identification of core-periphery structure in networks

Xiao Zhang, Travis Martin, and M. E. J. Newman

Phys. Rev. E **91**, 032803 — Published 6 March 2015

DOI: 10.1103/PhysRevE.91.032803

Identification of core-periphery structure in networks

Xiao Zhang,[1] Travis Martin,[2] and M. E. J. Newman[1,3]

[1]*Department of Physics, University of Michigan, Ann Arbor, MI 48109*
[2]*Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109*
[3]*Center for the Study of Complex Systems, University of Michigan, Ann Arbor, MI 48109*

Many networks can be usefully decomposed into a dense core plus an outlying, loosely-connected periphery. Here we propose an algorithm for performing such a decomposition on empirical network data using methods of statistical inference. Our method fits a generative model of core–periphery structure to observed data using a combination of an expectation–maximization algorithm for calculating the parameters of the model and a belief propagation algorithm for calculating the decomposition itself. We find the method to be efficient, scaling easily to networks with a million or more nodes and we test it on a range of networks, including real-world examples as well as computer-generated benchmarks, for which it successfully identifies known core–periphery structure with low error rate. We also demonstrate that the method is immune from the detectability transition observed in the related community detection problem, which prevents the detection of community structure when that structure is too weak. There is no such transition for core–periphery structure, which is detectable, albeit with some statistical error, no matter how weak it is.

I. INTRODUCTION

Much of the recent work on the structure of networked systems, such as social and technological networks, has focused on measurements of local structure, such as vertex degrees, clustering coefficients, correlations, and so forth [1, 2]. Increasingly, however, researchers have investigated medium- and large-scale structure as well. The lion's share of the attention has gone to the study of so-called community structure [3], the archetypal example of large-scale network structure, in which the nodes of a network are divided into tightly knit groups or communities that often reflect aspects of network function. But other structure types can be important as well and recent research has also looked at overlapping or fuzzy communities [4–6], hierarchical structure [7, 8], and ranking [9], among others.

In this paper we focus on another, distinct type of large-scale structure, *core–periphery structure*. Many networks are observed to divide into a densely interconnected core surrounded by a sparser halo or periphery. Already in the 1990s sociologists observed such structure in social networks [10] and more recently a number of researchers have made quantitative studies of core–periphery structure in a range of different types of networks [11–13]. The identification of core–periphery structure has a number of potential uses. Core nodes in a network might play a different role from periphery ones [14] and the ability to distinguish core from periphery might thus give us a new handle on function in networked systems. Distinguishing between core and periphery might lead to more informative visualizations of networks or find a role in graph layout algorithms similar to that played today by community structure. And core nodes, for instance in social networks, might be more influential or powerful than periphery ones, so the ability to discern the difference could shed light on social or other organization.

There have also been studies of some other types of structure that are reminiscent of, though different in important ways from, core–periphery structure: "rich club" structure [15, 16], degree assortativity [17, 18] and k-cores [2, 19]. A rich-club is a group of high-degree nodes in a network (i.e., nodes with many connections to others) that preferentially connect to one another. Such a club is a special case of the core in a core–periphery structure, but the concept of a core is more general, encompassing cases (as we will see) in which low-degree nodes can also belong to the core. The rich-club phenomenon also makes no statement about connectivity patterns in the remainder of the network, whereas core–periphery structure does.

Assortative mixing is the tendency of nodes in a network to connect to others that are similar to themselves in some way, and degree-assortative mixing is the tendency to connect to others with similar degree—high to high, and low to low. This produces a core of connected high-degree vertices similar to a rich club, but low-degree vertices also preferentially connect to one another and prefer not to connect to the core, which is the opposite of core–periphery structure as commonly understood, in which periphery vertices are more likely to connect to the core than they are to one another.

A k-core is a set of nodes in a network such that each has at least k connections to others in the set. A k-core with high k value is qualitatively similar to a rich club in being a set of interconnected high-degree vertices but, also like a rich club, it says nothing about connection patterns to the rest of the network. Moreover, again like a rich club, such a k-core consists only of high-degree vertices, whereas our definition of core–periphery structure allows for low-degree vertices to belong to the core as well.

A number of suggestions have been made about how, given the complete pattern of connections in a network, one could detect core–periphery structure. Many of them take the approach of defining an objective function that measures the strength or quality of a candidate division

into core and periphery and then maximizing (usually only approximately) over divisions to find the best one. In early work, Borgatti and Everett [10] proposed a quality function based on comparing the network to an ideal core–periphery model in which nodes are connected to each other if and only if they are members of the core. Rombach *et al.* [12] built on the same idea, but using a more flexible model. Holme [11] took a contrasting approach reminiscent of the clustering coefficient used to quantify transitivity in networks, while Lee *et al.* [20] made use of centrality measures based on notions of local density and transport in networks.

In this paper we propose a different, statistically principled method for detecting core–periphery structure using a maximum-likelihood fit to a generative network model. The method is conceptually similar to recently-popular first-principles methods for community detection [21, 22] and in fact uses the same underlying network model, the stochastic block model, although with a different choice of parameters appropriate to core–periphery rather than community structure. Among other results we demonstrate that the method is able consistently to detect planted core–periphery structure in computer-generated test networks, and that, by contrast with the community detection problem, there is no minimum amount of structure that can be detected. Any core–periphery structure, no matter how weak, is in principle detectable.

II. THE STOCHASTIC BLOCK MODEL

The stochastic block model is a well established and widely used model for community structure in networks. It is a generative model, meaning its original purpose is to create artificial networks that contain community structure. It is also commonly used, however, for community detection by fitting the model to observed network data. The parameters of the fit tell us the best division of the network into communities.

The model is defined as follows. We take n nodes, initially without any edges connecting them, and divide them into some number of groups. We will consider the simplest case where there are just two groups (which will represent the core and periphery). Each vertex is assigned randomly to group 1 with probability γ_1 or group 2 with probability $\gamma_2 = 1 - \gamma_1$. Then between every vertex pair we place an undirected edge independently at random with probability p_{rs}, or not with probability $1 - p_{rs}$, where r and s are the groups to which the two vertices belong. Thus the probability of connection of any two vertices depends solely on their group membership. The probabilities p_{rs} form a matrix, sometimes called the *mixing matrix* or *affinity matrix*, which is a 2×2 matrix in our two-group example. Since the edges in the network are undirected it follows that the mixing matrix is symmetric, $p_{12} = p_{21}$, leaving three independent probabilities that we can choose, p_{11}, p_{12}, and p_{22}.

In the most commonly studied case the probabilities for connection within groups are chosen to be larger than the probabilities between groups, so that p_{11}, p_{22} are both greater than p_{12}. This gives traditional community structure, also called assortative mixing, with denser connections within groups than between them. A contrasting possibility is the disassortative choice where p_{11}, p_{22} are smaller than p_{12}, so that edges are more probable between groups than within them. This choice, and the structure it describes, has received a modest amount of attention in the literature [23, 24].

There is, however, a third possibility that has rarely been studied, in which $p_{11} > p_{12} > p_{22}$. This is the situation we refer to as core–periphery structure. Since the group labels are arbitrary we can, without loss of generality, assume p_{11} to be the largest of the three probabilities, so group 1 is the core. Connections are most probable within the core, least probable within the periphery, and of intermediate probability between core and periphery. Note that this means that periphery vertices are more likely to be connected to core vertices than to each other, a characteristic feature of core–periphery structure that distinguishes it from either assortative or disassortative mixing.

As we have said, the stochastic block model can be used to detect structure in network data by fits of the data to the model. For instance, the assortative version of the model can be used to fit and hence detect community structure in networks [21, 22, 25, 26]. As shown in [27], however, it often performs poorly at this task in real-world situations because real-world networks tend to have broad degree distributions that dominate their large-scale structure and the fit tends to pick out this gross effect rather than the more subtle underlying community structure—typically the fit just ends up dividing the network into groups of higher- and lower-degree vertices rather than traditional communities. A more nuanced view has been given by Decelle *et al.* [28], who show that in fact both the degree-based division and the community division are good fits to the model—local maxima of the likelihood in the language introduced below—but the degree-based one is better.

But when we turn to core–periphery structure this bug becomes a feature. In networks with core–periphery structure the vertices in the core typically do have higher degree than those in the periphery, so a method that recognizes this fact is doing the right thing. Indeed, as we show in Section V, one can in certain cases do a reasonable job of detecting core–periphery structure just by separating vertices into two groups according to their degrees. On the other hand, one can do better still using the stochastic block model.

III. FITTING TO EMPIRICAL DATA

We propose to detect core–periphery structure in networks by finding the parameters of the stochastic block model that best fit the model to a given observed net-

work. This we do by the method of maximum likelihood, implemented using an expectation–maximization or EM algorithm [29, 30]. The use of EM algorithms for network model fitting is well established [31, 32], but it is worth briefly running through the derivation for our particular model, which goes as follows.

A. The EM algorithm

Given a network, the question we ask is, if this network were generated by the stochastic block model, what is our best guess at the values of the parameters of that model? To answer this question, let A_{ij} be an element of the adjacency matrix A of the network having value one if there is an edge between vertices i and j and zero otherwise, and let g_i be the group to which vertex i belongs. Then the probability, or likelihood, that the network was generated by the model, given the values of the model parameters p_{rs} and γ_r for all groups, is

$$\begin{aligned} P(A|p,\gamma) &= \sum_g P(A|p,g)P(g|\gamma) \\ &= \sum_g \prod_{i<j} p_{g_i g_j}^{A_{ij}} (1-p_{g_i g_j})^{1-A_{ij}} \prod_i \gamma_{g_i}, \end{aligned} \tag{1}$$

where $\sum_g$ indicates a sum over all assignments of the vertices to groups.

To determine the most likely values of the parameters p_{rs} and γ_r, we maximize this likelihood with respect to them. In fact it is technically simpler to maximize the logarithm of the likelihood:

$$\log P(A|p,\gamma) = \log \sum_g \prod_{i<j} p_{g_i g_j}^{A_{ij}} (1-p_{g_i g_j})^{1-A_{ij}} \prod_i \gamma_{g_i}, \tag{2}$$

which is equivalent since the logarithm is a monotone increasing function. Direct maximization is still quite difficult, however. Simply differentiating to find the maximum leads to a complex set of implicit equations that have no easy solution.

A better approach, and the one taken in the EM algorithm, involves the application of Jensen's inequality, which says that for any set of positive-definite quantities x_i

$$\log \sum_i x_i \geq \sum_i q_i \log \frac{x_i}{q_i}, \tag{3}$$

where q_i is any probability distribution satisfying the normalization condition $\sum_i q_i = 1$. One can easily verify that the exact equality is achieved by choosing

$$q_i = \frac{x_i}{\sum_i x_i}. \tag{4}$$

For any properly normalized probability distribution $q(g)$ over the group assignments g, Jensen's inequality applied to Eq. (2) gives

$$\begin{aligned} \log P(A|p,\gamma) &\geq \sum_g q(g) \log\biggl[\frac{1}{q(g)} \prod_{i<j} p_{g_i g_j}^{A_{ij}} (1-p_{g_i g_j})^{1-A_{ij}} \prod_i \gamma_{g_i}\biggr] \\ &= \sum_g q(g) \biggl[\sum_{i<j} \bigl[A_{ij} \log p_{g_i g_j} + (1-A_{ij}) \log(1-p_{g_i g_j})\bigr] + \sum_i \log \gamma_{g_i} - \log q(g)\biggr] \\ &= \frac{1}{2} \sum_{ij} \sum_{rs} \bigl[A_{ij} q_{rs}^{ij} \log p_{rs} + (1-A_{ij}) q_{rs}^{ij} \log(1-p_{rs})\bigr] + \sum_{ir} q_r^i \log \gamma_r - \sum_g q(g) \log q(g), \end{aligned} \tag{5}$$

where q_r^i is the *marginal probability* within the chosen distribution $q(g)$ that vertex i belongs to group r:

$$q_r^i = \sum_g q(g) \delta_{g_i,r}, \tag{6}$$

and q_{rs}^{ij} is the joint or two-vertex marginal probability that vertex i belongs to group r and vertex j simultaneously belongs to group s:

$$q_{rs}^{ij} = \sum_g q(g) \delta_{g_i,r} \delta_{g_j,s}, \tag{7}$$

with δ_{ij} being the Kronecker delta.

Following Eq. (4), the exact equality in (5) is achieved when

$$q(g) = \frac{\prod_{i<j} p_{g_i g_j}^{A_{ij}} (1-p_{g_i g_j})^{1-A_{ij}} \prod_i \gamma_{g_i}}{\sum_g \prod_{i<j} p_{g_i g_j}^{A_{ij}} (1-p_{g_i g_j})^{1-A_{ij}} \prod_i \gamma_{g_i}}. \tag{8}$$

Thus calculating the maximum of the left-hand side of (5) with respect to the parameters p, γ is equivalent to first maximizing the right-hand side with respect to $q(g)$ (by choosing the value above) so as to make the two sides equal, and then maximizing the result with respect to the parameters. In this way we turn our original problem of maximizing over the parameters into a double maximization of the right-hand side expression over the parameters

and the distribution $q(g)$. At first glance, this seems to make the problem more difficult, but numerically it is in fact easier, since it splits a challenging maximization into two separate and relatively elementary operations. The maximization with respect to the parameters is achieved by straightforward differentiation of (5) with the constraint that $\sum_r \gamma_r = 1$. Note that the final term on the right-hand side does not depend on the parameters and hence vanishes upon differentiation, and we arrive at the following expressions for the parameters:

$$p_{rs} = \frac{\sum_{ij} A_{ij} q_{rs}^{ij}}{\sum_{ij} q_{rs}^{ij}}, \tag{9}$$

$$\gamma_r = \frac{1}{n} \sum_i q_r^i, \tag{10}$$

where n is the total number of vertices as previously. The simultaneous solution of Eqs. (8) to (10) now gives us the optimal values of the parameters.

The EM algorithm solves these equations by numerical iteration. Given an initial guess at the parameters p and γ we can calculate the probability distribution $q(g)$ from Eq. (8) and from it the one- and two-vertex marginal probabilities, Eqs. (6) and (7). From these we then calculate a new estimate of p and γ from Eqs. (9) and (10). It can be proved that upon iteration this process will always converge to a local maximum of the log-likelihood [29]. It may not be the global maximum, however, so commonly one performs the entire calculation several times with different starting conditions, choosing from among the solutions so obtained the one with the highest likelihood.

Equation (9) can be simplified a little further by using Eq. (7) to rewrite the denominator thus:

$$\sum_{ij} q_{rs}^{ij} = \sum_g q(g) \sum_i \delta_{g_i,r} \sum_j \delta_{g_j,s} = \langle n_r n_s \rangle \,, \tag{11}$$

where $\langle \ldots \rangle$ indicates an average within the probability distribution $q(g)$ and $n_r = \sum_i \delta_{g_i,r}$ is the number of vertices in group r. In the limit of large network size, the number of vertices in a group becomes narrowly peaked and we can replace $\langle n_r n_s \rangle$ by $\langle n_r \rangle \langle n_s \rangle$ with

$$\langle n_r \rangle = \sum_g q(g) \sum_i \delta_{g_i,r} = \sum_i q_r^i, \tag{12}$$

where we have used Eq. (6). Then

$$p_{rs} = \frac{\sum_{ij} A_{ij} q_{rs}^{ij}}{\sum_i q_r^i \sum_j q_s^j}. \tag{13}$$

This expression has the advantage of requiring only a sum over edges in the numerator (since one need sum only those terms for which $A_{ij} = 1$) and single sums over vertices in the denominator, not the double sum in the denominator of (9). This makes evaluation of p_{rs} significantly faster for large networks. (Note, however, that despite appearances, Eq. (13) does not assume that $q_{rs}^{ij} = q_r^i q_s^j$, which would certainly not be correct in general. Only the sum over all vertex pairs factorizes, not the individual terms.)

The final output of the EM algorithm gives us not only the values of the parameters, but also the marginal probabilities q_r^i for vertices to belong to each group. In fact, it is normally this latter quantity that we are really interested in. In the community structure context it gives the probability that vertex i belongs to community r. In the core–periphery case, it gives the probability that the vertex belongs to either the core (group 1) or the periphery (group 2). Typically, the last step in the calculation is to assign each vertex to the group for which it has highest probability of membership, producing the final division of the network into core and periphery.

B. Belief propagation

The EM algorithm is an elegant approach but it has its shortcomings. Principal among them is the difficulty of performing the sum over group assignments g in the denominator of Eq. (8). Even for the current case where there are just two groups, this sum has 2^n terms and would take prohibitively long to perform numerically for any but the smallest of networks. The most common way around this problem is to make an approximate estimate of the sum by Monte Carlo sampling, but in this paper we employ an alternative technique proposed by Decelle *et al.* [22, 28], which uses belief propagation. This technique is of interest both because it is significantly faster than Monte Carlo and also because it lends itself to further analysis, as discussed in Section IV.

Belief propagation [33], a generalization of the Bethe–Peierls iterative method for the solution of mean-field models [34, 35], is a message-passing technique for finding probability distributions on networks, which we can use in this case to find the distribution $q(g)$ of Eq. (8). We define a "message" $\eta_r^{i\to j}$, which is equal to the probability that vertex i belongs to group r if vertex j is removed from the network. The removal of j allows one to derive a set of self-consistent of equations that must be satisfied by these messages [22, 36]. The equations are particularly simple for the case of a sparse network where p_{rs} is small so that terms of order p_{rs} can be ignored by comparison with terms of order 1, which appears to describe most real-world networks. For this case, the equations are

$$\eta_r^{i\to j} = \frac{\gamma_r}{Z_{i\to j}} \prod_{\substack{k \\ A_{ik}=0}} \Bigl[1 - \sum_s q_s^k p_{rs}\Bigr] \prod_{\substack{k(\neq j) \\ A_{ik}=1}} \sum_s \eta_s^{k\to i} p_{rs}, \tag{14}$$

where $Z_{i\to j}$ is a normalizing constant whose value is cho-

sen to ensure that $\sum_r \eta_r^{i\to j} = 1$ thus:

$$Z_{i\to j} = \sum_r \gamma_r \prod_{\substack{k\\A_{ik}=0}} \Big[1 - \sum_s q_s^k p_{rs}\Big] \prod_{\substack{k(\neq j)\\A_{ik}=1}} \sum_s \eta_s^{k\to i} p_{rs}. \tag{15}$$

Equation (14) is typically solved numerically, by starting from a random initial condition and iterating to convergence. In addition to calculating new values for the messages $\eta_r^{i\to j}$ on each step of this iteration we also need to calculate new values for the one-vertex marginal probabilities q_r^i, which satisfy

$$q_r^i = \frac{\gamma_r}{Z_i} \prod_{\substack{k\\A_{ik}=0}} \Big[1 - \sum_s q_s^k p_{rs}\Big] \prod_{\substack{k\\A_{ik}=1}} \sum_s \eta_s^{k\to i} p_{rs}, \tag{16}$$

with Z_i being another normalization constant:

$$Z_i = \sum_r \gamma_r \prod_{\substack{k\\A_{ik}=0}} \Big[1 - \sum_s q_s^k p_{rs}\Big] \prod_{\substack{k\\A_{ik}=1}} \sum_s \eta_s^{k\to i} p_{rs}. \tag{17}$$

Equation (14) is strictly true only on networks that are trees or are *locally tree-like*, meaning that in the limit of large network size the neighborhood of any vertex looks like a tree out to arbitrarily large distances. The stochastic block model itself generates networks that are locally tree-like, but many real-world networks are not, meaning that the belief-propagation method is only approximate in those cases. In practice, however, it appears to give good results, comparable in quality with those from Monte Carlo sampling [36] (which is also an approximate method).

Once the belief propagation equations have converged, we can use the results to evaluate Eq. (13). This requires values of the two-vertex marginals, which are given by Bayes theorem, to be

$$\begin{aligned} q_{rs}^{ij} &= P(g_i = r, g_j = s | A_{ij} = 1) \\ &= \frac{P(g_i = r, g_j = s)}{P(A_{ij} = 1)} P(A_{ij} = 1 | g_i = r, g_j = s), \end{aligned} \tag{18}$$

where all elements of the adjacency matrix other than A_{ij} are assumed given in each probability. In terms of our other variables we have

$$\begin{aligned} P(g_i = r, g_j = s) &= \eta_r^{i\to j} \eta_s^{j\to i}, \\ P(A_{ij} = 1 | g_i = r, g_j = s) &= p_{rs}, \end{aligned} \tag{19}$$

and the normalization $P(A_{ij} = 1)$ is fixed by the requirement that q_{rs}^{ij} sum to unity. So

$$q_{rs}^{ij} = \frac{\eta_r^{i\to j} \eta_s^{j\to i} p_{rs}}{\sum_{rs} \eta_r^{i\to j} \eta_s^{j\to i} p_{rs}}. \tag{20}$$

Substituting the values of q_r^i and q_{rs}^{ij} into Eqs. (10) and (13) then completes the EM algorithm.

Note that there are now two entirely separate iterative sections of our calculation: the EM algorithm, which consists of the iteration of Eqs. (8), (10), and (13), and the belief propagation algorithm, which consists of the iteration of Eq. (14).

Using the belief propagation algorithm is far faster than calculating $q(g)$ directly from Eq. (8). Equations (14) to (17) require the evaluation of only $\mathrm{O}(m+n)$ terms for a network with n vertices and m edges, meaning an iteration takes linear time in the common case of a sparse network with $m \propto n$. There is still the issue of how many iterations are needed for convergence, for which there are no firm results at present, but heuristic arguments suggest that the number of iterations needed is of order the diameter of a network, which is $\mathrm{O}(\log n)$ in typical real-world networks, so the algorithm is expected to converge in a small number of iterations.

The complete algorithm for detecting core–periphery structure in networks consists of the following steps:

1. Make an initial random guess at the values of the parameters p, γ.

2. From a random initial condition, iterate to convergence the belief propagation equations (14) for vertex pairs connected by an edge and the one-vertex marginal probabilities, Eq. (16).

3. Use the converged values to calculate the two-vertex marginal probabilities, Eq. (20).

4. Use the one- and two-vertex probabilities to calculate an improved estimate of the parameters from Eqs. (10) and (13).

5. Repeat from step 2 until the parameters converge.

6. Assign each vertex to either the core or the periphery, whichever has the higher probability q_r^i.

IV. DETECTABILITY

One of the most intriguing aspects of the community detection problem is the *detectability threshold* [22, 37, 38]. When a network contains strong community structure—when there is a clear difference in density between the in-group and out-group connections—then that structure is easy to detect and a wide range of algorithms will do a good job. When structure becomes sufficiently weak, however, at least in simple models of the problem such as the stochastic block model, it becomes undetectable. In this weak-structure regime it is rigorously provable that no algorithm can assign nodes to communities with success any better than a random coin toss [39, 40]. Given the strong connection between community detection and the core–periphery problem studied here, it is natural to ask whether there is a similar threshold for core–periphery detection. Is there a point

at which core–periphery structure becomes so weak as to be undetectable by our method or any other?

At the most naive level, the answer to this question is no. The core–periphery problem differs from the community detection problem in that the vertices in the core have higher degree on average than those in the periphery and hence one can use the degrees to identify the core and periphery vertices with an average success rate better than a coin toss.

Consider in particular the common case of a stochastic block model where

$$p_{rs} = \frac{c_{rs}}{n} \tag{21}$$

for some constants c_{rs}. This is the case for which the detectability threshold mentioned above is observed. Then the average degrees in the core and periphery are, respectively,

$$\bar{d}_1 = \gamma_1 c_{11} + \gamma_2 c_{12}, \qquad \bar{d}_2 = \gamma_1 c_{12} + \gamma_2 c_{22}, \tag{22}$$

and the difference is $\bar{d}_1 - \bar{d}_2 = \gamma_1(c_{11} - c_{12}) + \gamma_2(c_{12} - c_{22})$. Since, by hypothesis, $c_{11} > c_{12} > c_{22}$, this quantity is always positive and $\bar{d}_1 > \bar{d}_2$. Because the edges in the network are independent, the actual degrees have a Poisson distribution about the mean in the limit of large n, and hence the degree distribution consists of two overlapping Poisson distributions, as sketched in Fig. 1. By simply dividing the vertices according to their observed degrees, therefore, we can (on average) classify them as core or periphery with success better than chance. (This assumes we know the sizes of the two groups, which we usually don't, but this problem can be solved—see Section IV A.)

So rather than asking whether our ability to detect structure fails completely in the weak-structure limit, we should instead ask whether we can do any better than simply dividing vertices according to degree. The answer to this question is both yes and no. As we now show, in the limit of weak structure no algorithm can do better than one that looks at degrees only, but for stronger structure we can do better in most cases.

To demonstrate these results, we take a standard approach from statistics and ask whether our detection algorithm based on the stochastic block model can detect core–periphery structure in networks that are themselves generated using the stochastic block model. This is a so-called consistency test and, in addition to providing a well-controlled test of the algorithm, it has one very important advantage. It is known that on average the best way to detect the structure in a data set generated by a model is to perform a maximum-likelihood fit to that same model, exactly as our algorithm does. No other algorithm will return better performance on this test, on average, than the maximum likelihood method.

Bearing this in mind, consider applying the algorithm of this paper to a network generated using the stochastic block model with two equally sized groups ($\gamma_1 = \gamma_2 = \frac{1}{2}$) and weak core–periphery structure of the form

$$c_{11} = c + \alpha_1\delta, \quad c_{12} = c, \quad c_{22} = c - \alpha_2\delta, \tag{23}$$

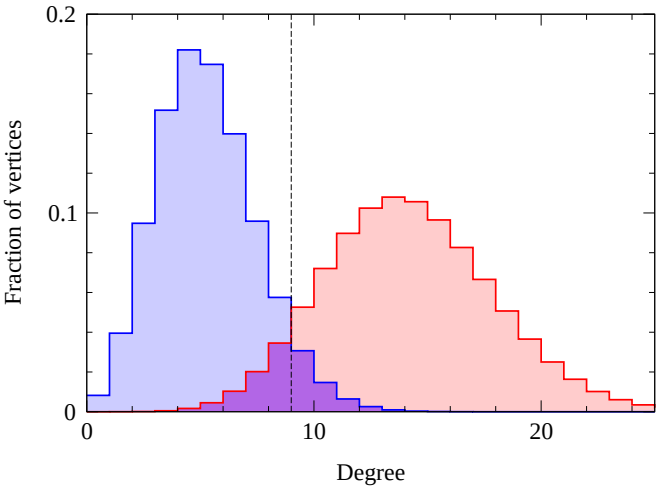

FIG. 1: In the stochastic block model both core (red) and periphery (blue) vertices have Poisson degree distributions, but the mean degree is higher in the core than in the periphery, so the overall degree distribution of the network is a sum of two overlapping Poisson distributions as shown here. A simple division of vertices by degree (vertical dashed line) classifies most vertices into the correct groups, red in the core and blue in the periphery. Only those in the overlap (shown in purple) are classified incorrectly.

where α_1, α_2, and c are O(1) positive constants and δ is a small quantity. In the limit as $\delta \to 0$ the core–periphery structure vanishes and the network becomes a uniform random graph of average degree c. For small values of δ the structure is weak and it is this regime that we are interested to probe.

To make the problem as simple as possible, suppose that we allow our algorithm to use the exact values of the parameters γ_r and p_{rs}, meaning that we need only perform the belief propagation part of the calculation to derive an answer. There is no need to perform the EM algorithm iteration as well, since this is only needed to determine the parameters. This is a somewhat unrealistic situation—in practical cases we do not normally know the values of the parameters. However, if, as we will show, the algorithm performs poorly in this situation then it will surely perform no better if we give it *less* information—if we do not know the values of the parameters. Thus this choice gives us a best-case estimate of the performance of the algorithm.

To gain a theoretical understanding of how the belief propagation process works, we consider the *odds ratio* q_1^i/q_2^i between the probabilities that a vertex belongs to the core and the periphery. Making use of Eq. (16), expanding the first product to leading order in $p_{rs} = c_{rs}/n$, and dividing top and bottom in the second product by a factor of n, this quantity is given by

$$\frac{q_1^i}{q_2^i} = \frac{\gamma_1}{\gamma_2}\,\mathrm{e}^{\bar{d}_2 - \bar{d}_1} \prod_{\substack{k \\ A_{ik}=1}} \frac{\eta_1^{i\to j} c_{11} + \eta_2^{i\to j} c_{12}}{\eta_1^{i\to j} c_{12} + \eta_1^{i\to j} c_{22}}, \tag{24}$$

where $\bar{d}_1$ and $\bar{d}_2$ are defined as in Eq. (22) and we have made use of Eq. (10). Note how the normalization $Z_{i\to j}$ cancels, making calculations simpler.

Now we substitute for c_{rs} from Eq. (23), set $\gamma_1 = \gamma_2 = \frac{1}{2}$, and note that as $\delta \to 0$ the probabilities of any vertex being in one group or the other become equal, so that

$$\frac{\eta_1^{i\to j}}{\eta_2^{i\to j}} = 1 + \beta_{i\to j}\delta \tag{25}$$

to leading order for some constant $\beta_{i\to j}$. Keeping terms to first order in δ, we then find that

$$\frac{q_1^i}{q_2^i} = 1 + \tfrac{1}{2}(\alpha_1 + \alpha_2)\frac{k_i - c}{c}\delta, \tag{26}$$

where k_i is the degree of vertex i as previously.

Note that $\beta_{i\to j}$ has dropped out of this expression, meaning that when δ is small and the structure is weak the probabilities depend only on the degree k_i of the vertex and not on any other properties of the network structure. More specifically, vertex i has a higher probability of belonging to group 1, i.e., the core, whenever its degree k_i is greater than the average degree c in the network as a whole. When its degree is below average the vertex has a higher probability of belonging to the periphery. Thus a simple division based on probabilities is precisely equivalent to dividing based on degree. Moreover, since, as we have said, no other algorithm can do better at distinguishing the structure, it immediately follows that there is nothing better one can do in the weak-structure limit than divide the vertices based on degree.

The same is also true in the limit of strong structure. If the core–periphery structure is strong, meaning that there is a big difference between connection probabilities for core and periphery vertices, then the two Poisson distributions of Fig. 1 will be far apart, with very little overlap, and vertices can be accurately classified by degree alone. The means of the two distributions are $\mu_1 = \frac{1}{2}(c_{11} + c_{12})$ and $\mu_2 = \frac{1}{2}(c_{12} + c_{22})$ and, since the width of a Poisson distribution scales as the square root of its mean, we will have easily distinguishable peaks provided $\mu_1 - \mu_2 \gg \sqrt{(\mu_1 + \mu_2)/2}$, or

$$c_{11} - c_{22} \gg 2\sqrt{c}, \tag{27}$$

where $c = \frac{1}{2}(\mu_1 + \mu_2)$ is the average degree of the network as a whole.

In fact, even between the limits of strong and weak structure there are some networks for which a simple division by degrees is optimal. Consider the two-parameter family of models defined by

$$c_{11} = \theta r, \quad c_{12} = \theta, \quad c_{22} = \frac{\theta}{r}, \tag{28}$$

for any choice of γ_r, where $\theta > 0$ and $r > 1$. Substituting this choice into Eq. (24) gives

$$\frac{q_1^i}{q_2^i} = \frac{\gamma_1}{\gamma_2}\,\mathrm{e}^{\bar{d}_2 - \bar{d}_1} r^{k_i}, \tag{29}$$

so again the results depend only on the vertex degrees.

So are there any cases where we can do better than the algorithm that looks at degrees only? The answer is yes: for structure of intermediate strength, neither exceptionally weak nor exceptionally strong, and away from the plane in parameter space defined by Eq. (28), the messages are not simple functions of degree but depend in general on the details of the network structure. Since, once again, the belief propagation algorithm is optimal, it follows that any algorithm that gives a result different from the belief propagation algorithm must give an inferior one, including an algorithm that looks at degrees only. Hence in this regime one can do better than simply looking at vertex degrees. Moreover, this regime contains most cases of real-world interest. After all, core–periphery structure so weak as to be barely detectable is presumably not of great interest, and real-world networks rarely have strongly bimodal degree distributions of the kind considered above that make degree-based algorithms work well in the strong-structure limit.

There is also, we note, no evidence in this case of a detectability threshold or similar sharp discontinuity in the behavior of the algorithm. Everywhere in the parameter space the algorithm can identify core and periphery with performance better than chance.

A. Degree-based algorithm

We are now also in a position to answer a question raised parenthetically in Section III B. If we choose to classify vertices based on degree alone, what size groups should we use? We can answer this question by noting that Eq. (28) defines the subset of stochastic block models for which degree alone governs classification. As we have seen, fitting to this model is equivalent to dividing according to degree, but performing such a fit using the full EM algorithm, rather than just looking at degrees, has the added advantage that it gives us the values of the parameters γ_r, which in turn give us the expected sizes $n_r = n\gamma_r$ of the groups. We can perform the fit exactly as we did for the full stochastic block model in Section III A. Substituting Eq. (28) into the right-hand side of (5), differentiating, and neglecting terms of order $1/n$ by comparison with those of order 1, we find the optimal values of the parameters to be

$$\gamma_r = \frac{1}{n}\sum_i q_r^i, \qquad r = \frac{\kappa_1}{\kappa_2}, \qquad \theta = \frac{\kappa_1\kappa_2}{c}, \tag{30}$$

where c is the average degree of the network as previously and κ_r is the expected degree in group r:

$$\kappa_r = \frac{\sum_i k_i q_r^i}{\sum_i q_r^i}. \tag{31}$$

The one-vertex probabilities q_r^i are given by Eq. (29) to be

$$q_1^i = \frac{\gamma_1 \mathrm{e}^{-\bar{d}_1} r^{k_i}}{\gamma_2 \mathrm{e}^{-\bar{d}_2} + \gamma_1 \mathrm{e}^{-\bar{d}_1} r^{k_i}}, \qquad q_2^i = 1 - q_1^i. \tag{32}$$

Hence for this model, no belief propagation is necessary. One can simply iterate Eqs. (30) and (32) to convergence to determine the group memberships. (Note that in fact the parameter θ is never needed in the iteration—it is sufficient to calculate only γ_1, γ_2, and r from Eq. (30).)

V. APPLICATIONS AND PERFORMANCE

We have tested the proposed method on both computer-generated and real-world example networks.

A. Computer-generated test networks

Computer-generated networks provide a controlled test of the algorithm's ability to detect known structure. For these tests we make use of the stochastic block model itself to generate the test networks. We parametrize the mixing matrix of the model as

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \theta_1 \mathbf{u}_1 \mathbf{u}_1^T + \theta_2 \mathbf{u}_2 \mathbf{u}_2^T, \tag{33}$$

where $\mathbf{u}_1 = (\sqrt{r}, 1/\sqrt{r})$ and $\mathbf{u}_2 = (1/\sqrt{r}, -\sqrt{r})$. With this parametrization, setting $\theta_2 = 0$ recovers the (θ, r)-model of Section IV, for which, as we showed there, no algorithm does any better than a naive division according to vertex degree only. The parameter θ_2 measures how far away we are from that model in the perpendicular direction defined by $\mathbf{u}_2$, and we might guess that when we are further away—i.e., for values of θ_2 further from zero—we would see a greater difference between the belief propagation algorithm and the naive one.

Figure 2 shows this indeed to be the case. The figure shows, for three different choices of θ_1, the error rate of the algorithm (i.e., the fraction of incorrectly identified vertices) as a function of θ_2 for networks of $n = 1\,000\,000$ nodes, divided into equally sized core and periphery. Also shown on the plot is the performance of the algorithm that simply divides the vertices into two equally sized groups according to degree. As we can see, when $\theta_2 = 0$ (marked by the vertical dashed line) the results for the two approaches coincide as we expect. But as θ_2 moves away from zero there is a visible difference between the two, with the error rate of the naive algorithm being worse than that of belief propagation by a factor of ten or more in some cases.

It is fair to say, however, that the error rates of the two algorithms are comparable in some cases and the naive algorithm does moderately well under the right conditions, with error rates of around 10 or 20 percent for many choices of parameter values. There are a couple of

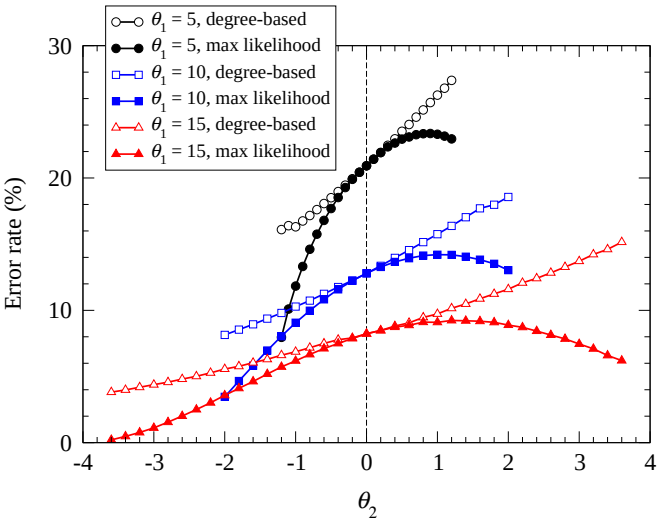

FIG. 2: The fraction of nodes classified incorrectly in tests on stochastic block model networks parametrized according to Eq. (33), as a function of θ_2 for fixed $r = 2$ and three different values of θ_1 as indicated. Solid points represent results for the maximum likelihood method described in this paper. Open points are the results of a simple division according to vertex degree. Each point is an average over 10 networks of a million nodes each. Statistical errors are smaller than the data points. The parameter ranges are different for different curves because they are constrained by the requirement that edge probabilities be nonnegative and that $c_{11} > c_{12} > c_{22}$, which means that θ_2 must satisfy $-\theta_1/r < \theta_2 < \theta_1(r - 1/r)$.

possible morals one can derive from this observation. On the one hand, if one is not greatly concerned with accuracy and just wants a quick-and-dirty division into core and periphery, then dividing vertices by degree may be a viable strategy. The belief propagation method usually does better, but it is also more work to program and requires more CPU time to execute. For some applications we may feel that the additional effort is not worth the payoff. Moreover, since the belief propagation method is optimal in the sense discussed earlier, we know that, at least for the definition of core–periphery structure used here, no other algorithm will out-perform it, so the loss of accuracy seen in Fig. 2 is the largest such loss we will ever incur when using the degree-based algorithm. In other words, this is as bad as it gets, and it's not that bad.

On the other hand, as we have said, one does not in most cases know the sizes of the groups into which the network is to be divided, in which case one must use the EM algorithm even for a degree-based division. The computations involved, which are described in Section IV A, are less arduous than those for the full belief propagation algorithm but significantly more complex than a simple division by degree only, and this eliminates some of the advantages of the degree-based approach.

Furthermore, while the number of nodes on which our

method and the degree-based algorithm differ is sometimes quite small, it may be these very nodes that are of greatest interest. It's true that it is typically the higher-degree nodes that fall in the core and the lower-degree ones that fall in the periphery. But when the two algorithms differ in their predictions it is precisely because some of the low-degree nodes correctly belong in the core or some of the high-degree ones in the periphery, which could lead us to ask what is special about these nodes. Who are the people in a social network, for example, who fall in the core even though they don't have many connections? Who are the well-connected people who fall in the periphery? These people may be of particular interest to us, but they can only be identified by using the full maximum likelihood algorithm. The degree-based algorithm will, by definition, fail to find these anomalous nodes.

B. Real-world examples

Figure 3 shows an application of our method to a real-world network, the Internet, represented at the level of autonomous systems. This network is expected to have clear core–periphery structure: its general structure consists of a large number of leaves or edge nodes—typically client autonomous systems corresponding to end users like ISPs, corporations, or educational institutions—plus a smaller number of well-connected backbone nodes [11, 41]. This structure is reflected in the decomposition discovered by our analysis, indicated by the blue (core) and yellow (periphery) nodes in the figure. The bulk of the nodes are placed in the periphery, while a small fraction of central hubs are placed in the core. Note, however, that, as discussed earlier, the algorithm does not simply divide the nodes according to degree. There are a significant number of high-degree nodes that are placed by the algorithm in the periphery because of their position on the fringes of the network, even though their degree might naively suggest that they be placed in the core.

Figure 4 shows a contrasting example. The network in this figure, drawn from a 2005 study by Adamic and Glance [42] is a web network, representing a set of 1225 weblogs, personal commentary web sites, devoted in this case to commentary on US politics. Edges represent hyperlinks between blogs, which we treat as undirected for the purposes of our analysis. This network has been studied previously as an example of community structure, since it displays a marked division into groups of conservative and liberal blogs. The figure is drawn so as to make these groups clear to the eye—they correspond roughly to the left and right halves of the picture—and the core–periphery division is indicated once more by the blue (core) and yellow (periphery) nodes.

As the figure shows, the analysis finds a clear separation between core and periphery, and moreover finds a separate core in each of the two communities. In effect, the conservative blogs are divided into a conservative core and periphery, and similarly for the liberal ones. A direct examination of the list of core nodes in each community finds them to contain, as we might expect, many prominent blogs on either side of the aisle, such as the National Review and Red State on the conservative side and Daily Kos and Talking Points Memo on the liberal side.

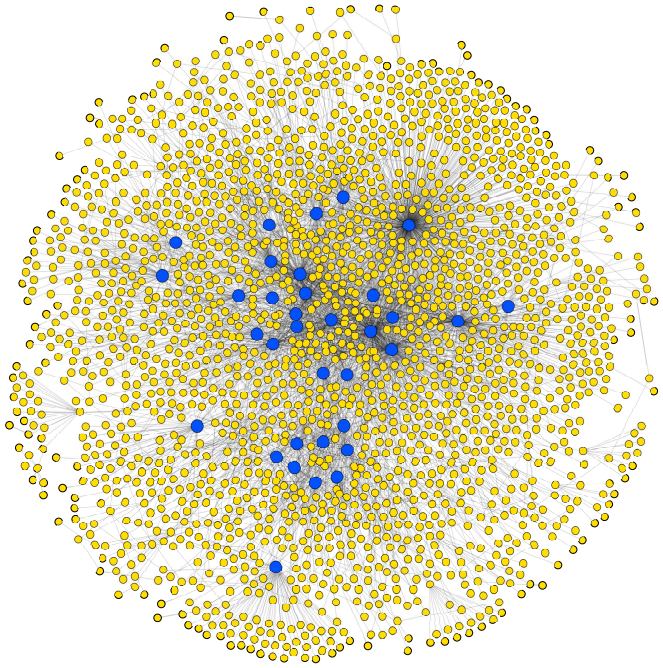

FIG. 3: Core–periphery division of a 1470-node representation of the Internet at the level of autonomous systems [17]. Nodes placed in the core by our analysis are drawn larger and in blue; nodes in the periphery are smaller and in yellow. The network was constructed from data from the Oregon Route-views Project and represents an older snapshot, chosen for the network's relatively small size. Our methods can easily be applied to larger networks, but the results are harder to visualize in an informative fashion.

VI. CONCLUSION

We have examined core–periphery structure in undirected networks, proposing a first-principles algorithm for identifying such structure by fitting a stochastic block model to observed network data using a maximum likelihood method. The maximization is implemented using a combination of an expectation–maximization algorithm and belief propagation. The algorithm gives good results on test networks and is efficient enough to scale to networks of a million nodes or more. By a linearization of the belief propagation equations we are also able to show the method to be immune from the detectability threshold seen in the application of similar methods to community detection. In the community detection case the algorithm (and indeed all algorithms) fail when community structure in the network is too weak, but there is no such failure for the core–periphery case. Core–periphery

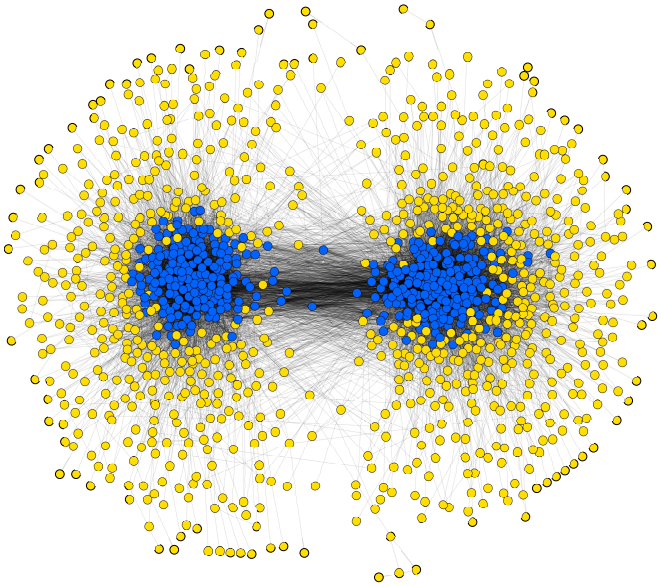

FIG. 4: Core–periphery division of a network of hyperlinks between political blogs taken from [42]. The network naturally separates into conservative and liberal communities, clearly visible as the two clusters in this picture. Within each group our algorithm finds a separate core and periphery indicated by the blue and yellow nodes respectively.

structure is always detectable, no matter how weak it is.

There are many questions are not answered by the current work. For example, it is an open question how one would perform a similar calculation on a weighted network. And one might reasonably generalize our calculation to more than two groups—the two-group core-periphery division is traditional, but one can envisage an onion-like division into cores within cores, with three, four, or more groups of vertices. We believe a generalization of this kind using the methods developed here would be straightforward, and would be a suitable topic for future research.

Acknowledgments

The authors thank Cris Moore for useful conversations and Petter Holme for providing the network data for Fig. 3. This work was funded in part by the National Science Foundation under grants DMS–1107796 and DMS–1407207 and by the Air Force Office of Scientific Research (AFOSR) and the Defense Advanced Research Projects Agency (DARPA) under grant FA9550–12–1–0432.

[1] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, Physics Reports **424**, 175 (2006).
[2] M. E. J. Newman, *Networks: An Introduction* (Oxford University Press, Oxford, 2010).
[3] S. Fortunato, Phys. Rep. **486**, 75 (2010).
[4] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Nature **435**, 814 (2005).
[5] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, Journal of Machine Learning Research **9**, 1981 (2008).
[6] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, Nature **466**, 761764 (2010).
[7] E. Ravasz and A.-L. Barabási, Phys. Rev. E **67**, 026112 (2003).
[8] A. Clauset, C. Moore, and M. E. J. Newman, Nature **453**, 98 (2008).
[9] B. Ball and M. E. J. Newman, Network Science **1**, 16 (2013).
[10] S. P. Borgatti and M. G. Everett, Social Networks **21**, 375 (1999).
[11] P. Holme, Phys. Rev. E **72**, 046111 (2005).
[12] M. P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha, SIAM J. Appl. Math. **74**, 167 (2014).
[13] P. Csermely, A. London, L.-Y. Wu, and B. Uzzi, Journal of Complex Networks **1**, xx (2013).
[14] R. Guimerà and L. A. N. Amaral, Nature **433**, 895 (2005).
[15] V. Colizza, A. Flammini, M. A. Serrano, and A. Vespignani, Nature Physics **2**, 110 (2006).
[16] S. Zhou and R. J. Mondragon, IEEE Comm. Lett. **8**, 180 (2004).
[17] R. Pastor-Satorras, A. Vázquez, and A. Vespignani, Phys. Rev. Lett. **87**, 258701 (2001).
[18] M. E. J. Newman, Phys. Rev. Lett. **89**, 208701 (2002).
[19] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes, Phys. Rev. Lett. **96**, 040601 (2006).
[20] S. H. Lee, M. Cucuringu, , and M. A. Porter, Phys. Rev. E **89**, 032810 (2014).
[21] P. J. Bickel and A. Chen, Proc. Natl. Acad. Sci. USA **106**, 21068 (2009).
[22] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, Phys. Rev. Lett. **107**, 065701 (2011).
[23] M. E. J. Newman, Phys. Rev. E **67**, 026126 (2003).
[24] X. Yan, Y. Zhu, J.-B. Rouquier, and C. Moore, in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Association of Computing Machinery, New York, 2011).
[25] T. P. Peixoto, Phys. Rev. Lett. **110**, 148701 (2013).
[26] T. P. Peixoto, Phys. Rev. X **4**, 011047 (2014).
[27] B. Karrer and M. E. J. Newman, Phys. Rev. E **83**, 016107 (2011).
[28] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, Phys. Rev. E **84**, 066106 (2011).
[29] A. P. Dempster, N. M. Laird, and D. B. Rubin, J. R. Statist. Soc. B **39**, 185 (1977).
[30] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, 2nd ed. (Wiley-Interscience, New York, 2008).
[31] K. Nowicki and T. A. B. Snijders, J. Amer. Stat. Assoc. **96**, 1077 (2001).
[32] M. E. J. Newman and E. A. Leicht, Proc. Natl. Acad.

Sci. USA **104**, 9564 (2007).
[33] J. Pearl, *Probabilistic Reasoning in Intelligent Systems* (Morgan Kaufmann, San Francisco, CA, 1988).
[34] H. A. Bethe, Proc. R. Soc. London A **150**, 552 (1935).
[35] R. Peierls, Cambridge Philos. Soc. B **2**, 477 (1936).
[36] X. Yan, C. R. Shalizi, J. E. Jensen, F. Krzakala, C. Moore, L. Zdeborova, P. Zhang, and Y. Zhu, J. Stat. Mech. P05007 (2014).
[37] J. Reichardt and M. Leone, Phys. Rev. Lett. **101**, 078701 (2008).
[38] D. Hu, P. Ronhovde, and Z. Nussinov, Phil. Mag. **92**, 406 (2012).
[39] E. Mossel, J. Neeman, and A. Sly, *Stochastic block models and reconstruction*, Preprint arxiv:1202.1499 (2012).
[40] E. Mossel, J. Neeman, and A. Sly, *A proof of the block model threshold conjecture*, Preprint arxiv:1311.4115 (2013).
[41] R. Pastor-Satorras and A. Vespignani, *Evolution and Structure of the Internet* (Cambridge University Press, Cambridge, 2004).
[42] L. A. Adamic and N. Glance, in *Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem* (2005).