Efficient method for estimating the number of communities in a network

Maria A. Riolo,[1] George T. Cantwell,[2] Gesine Reinert,[3] and M. E. J. Newman[1, 2]

[1] *Center for the Study of Complex Systems, University of Michigan, Ann Arbor, Michigan, USA*
[2] *Department of Physics, University of Michigan, Ann Arbor, Michigan, USA*
[3] *Department of Statistics, University of Oxford, 24–29 St. Giles, Oxford, UK*

While there exist a wide range of effective methods for community detection in networks, most of them require one to know in advance how many communities one is looking for. Here we present a method for estimating the number of communities in a network using a combination of Bayesian inference with a novel prior and an efficient Monte Carlo sampling scheme. We test the method extensively on both real and computer-generated networks, showing that it performs accurately and consistently, even in cases where groups are widely varying in size or structure.

I. INTRODUCTION

Many networks of interest in the sciences display community structure, meaning that their nodes divide naturally into clusters, modules, or groups, such that there are many network connections within groups but few between groups [1–3]. The decomposition of networks into their constituent communities is one of the primary tools used for interpreting the structure of large networked systems, allowing us to break data sets apart into manageable pieces and hence make sense of systems that can otherwise defy analysis.

Community detection—the process of identifying good community divisions of a given network—has been the subject of a vigorous research effort in the last 15 years or so, and many different approaches have been proposed. A fundamental shortcoming of most of them, however, is that they require us to know in advance how many communities a network contains. We don't usually know this number *a priori*, meaning we need some way to estimate it from the data. Recently, a number of authors, including ourselves, have proposed methods for making such estimates using Bayesian inference applied to fits of network models to observed network data [4–12]. In these approaches, one defines a generative random-graph model of a network with community structure, then fits it to the data to obtain a Bayesian posterior probability distribution over possible divisions of the network into groups, along with the associated number of groups, which we denote k. Then one averages over this distribution in some way to produce an estimate of the relative probability of different values of k for the network in question.

In this paper, which builds on our previous work in [10], we do a number of things. First, we give a detailed derivation of a practical method for computing the number of groups or communities in real-world network data. Many of the previous approaches have developed useful formal ideas, but not practical algorithms for real data, because they are based on unrealistic network models—most often the so-called stochastic block model, which is known to be a poor model for calculations of this kind [13]. In this paper we employ a more sophisticated model, the degree-corrected stochastic block model, which gives substantially superior results.

Second, we look carefully at the prior probability distribution over divisions of a network into groups, and particularly at generative processes for priors with non-empty groups. As in many Bayesian approaches, the choice of prior turns out to be crucial to performing useful inference, and in particular we point out that various types of uniform (maximum-entropy) priors, including ones used in previous work, give poor results and should be avoided. We propose a new prior based on a queueing-type process that appears to give excellent results in our tests.

Third, we describe an efficient Monte Carlo algorithm that exploits specific features of our proposed prior to perform rapid calculations on large networks.

Finally, we give the results of extensive tests of our method on both real and synthetic benchmark networks, which indicate that the method is able to consistently recover known values for the number of communities under real-world conditions.

II. DEGREE-CORRECTED STOCHASTIC BLOCK MODEL

The method we propose for estimating the number of communities in a network is based on techniques of statistical inference, in which observed network data are fitted to a generative model of network structure. The parameters of the fit tell us about the aggregate properties of the data in much the same way that the fit of a straight line through a set of data points can tell us about their slope.

The network model we employ in our calculations is the *degree-corrected stochastic block model* [13]. The traditional (non-degree-corrected) stochastic block model, first proposed in the 1980s [14], is a simple model for networks with community structure that has been widely studied in the statistics, sociology, and physics literature. Recently, however, it has been recognized that this model has serious shortcomings [13] because the networks it generates have a Poisson degree distribution within each community, making them very unlike most empirically observed networks, which typically have highly right-

skewed degree distributions. In practice, this means that the model is often unable to fit observed network data well for any choice of parameter values. The degree-corrected stochastic block model remedies this problem by introducing additional mechanisms that allow for arbitrary, non-Poisson degree distributions and is found in practice to give much better fits to real-world network data.

In the degree-correct stochastic block model n nodes are divided into some number k of groups, labeled $1 \ldots k$, with g_i denoting the group to which node i is assigned. Then edges are placed between nodes with probabilities that depend on group membership. There are several variants of the model in use that employ slightly different strategies for placing edges, but the most common strategy, and the one we use here, is to place between any node pair i, j a number a_{ij} of edges that is Poisson distributed with mean $\theta_i\theta_j\omega_{g_ig_j}$, where $\{\theta_i\}$ and $\{\omega_{rs}\}$ are sets of parameters whose values we choose. The numbers of edges a_{ij} form the elements of the adjacency matrix A of the network, the standard mathematical representation of network structure. Although these numbers are Poisson distributed, the expected degrees of the nodes can follow any distribution—within each group they are proportional to the values of the parameters θ_i, which we are at liberty to choose in any way we please.

The use of a Poisson distribution for the numbers of edges means that the network generated can in theory have multiedges, i.e., pairs of nodes connected by more than one edge, which is not usually realistic—most real-world networks do not have multiedges. Most networks of scientific interest, however, are very sparse, meaning that the values of $\theta_i\theta_j\omega_{g_ig_j}$ are very small and the probability of having two edges between the same pair of nodes is smaller still. Multiedges are, as a result, few enough in number that they can usually be neglected. One also normally allows self-edges in the network (edges that connect a node to itself), placing at each node i a Poisson distributed number of self-edges with mean $\frac{1}{2}\theta_i^2\omega_{g_ig_i}$. Again this is not realistic but in practice the number of self-edges is small, so they can be neglected. (By convention, the number of self-edges at node i is denoted $\frac{1}{2}a_{ii}$ and not a_{ii}, i.e., a_{ii} is twice the number of self-edges. The factor of $\frac{1}{2}$ in the number of self-edges $\frac{1}{2}\theta_i^2\omega_{g_ig_i}$ is included for consistency with this definition—it makes the expected value of a_{ij} equal to $\theta_i\theta_j\omega_{g_ig_j}$ for all i, j.)

The description above does not completely specify the degree-corrected block model because there remains an arbitrary normalization of the parameters θ_i that has yet to be fixed. We can increase the values of all the θ_i in group r by any factor we please and, provided we simultaneously decrease ω_{rs} by the same factor, the value of $\theta_i\theta_j\omega_{g_ig_j}$, and hence also the model itself, will not change. We can fix the values of the parameters by choosing a specific normalization for the θ_i. A number of choices are possible, all of which are ultimately equivalent, but for our purposes here a convenient choice is to fix the mean of the θ_i to be 1 in each group:

$$\frac{1}{n_r}\sum_{i=1}^{n}\theta_i\delta_{r,g_i} = 1, \tag{1}$$

where δ_{ij} is the Kronecker delta and $n_r = \sum_i \delta_{r,g_i}$ is the number of nodes in group r.

For the case of given number of group k and group assignments g this completes the specification of the model. With the model specified we can now write down the probability that any particular network with adjacency matrix $A = \{a_{ij}\}$ is generated:

$$\begin{aligned} P(A|\omega,\theta,g,k) &= \prod_{i<j}\bigl(\theta_i\theta_j\omega_{g_ig_j}\bigr)^{a_{ij}} e^{-\theta_i\theta_j\omega_{g_ig_j}} \\ &\qquad \times \prod_i \bigl(\tfrac{1}{2}\theta_i^2\omega_{g_ig_i}\bigr)^{a_{ii}/2} e^{-\theta_i^2\omega_{g_ig_i}/2} \\ &= \prod_i \theta_i^{d_i} \prod_{r<s}\omega_{rs}^{m_{rs}} e^{-n_rn_s\omega_{rs}} \prod_r \omega_{rr}^{m_{rr}} e^{-n_r^2\omega_{rr}/2}, \end{aligned} \tag{2}$$

where we have made use of Eq. (1) in the second equality, $d_i = \sum_j a_{ij}$ is the observed degree of node i, and

$$m_{rs} = \begin{cases} \sum_{ij} a_{ij}\delta_{g_i,r}\delta_{g_j,s} & \text{when } r \neq s, \\ \frac{1}{2}\sum_{ij} a_{ij}\delta_{g_i,r}\delta_{g_j,r} & \text{when } r = s, \end{cases} \tag{3}$$

is the number of edges running between groups r and s. We have also discarded an overall multiplying constant in Eq. (2), which has no effect on our results.

The parameters θ and ω are irrelevant to the questions we are interested in and can be integrated out. To do this we need to fix the prior probabilities on the parameters θ and ω. We assume the priors to be independent (conditioned on g, k), so that $P(\omega,\theta|g,k) = P(\omega|k)P(\theta|g,k)$, and

$$P(A|g,k) = \iint P(A|\omega,\theta,g,k)P(\theta|g,k)P(\omega|k)\,\mathrm{d}\theta\,\mathrm{d}\omega. \tag{4}$$

We employ maximum-entropy (i.e., least informative) priors on both θ and ω. For θ this means a uniform prior over the regular simplex of values specified by Eq. (1). For ω the situation is more complex. We note that the expected number of edges between groups r and s is

$$\begin{aligned} \sum_{ij}\theta_i\theta_j\omega_{g_ig_j}\delta_{r,g_i}\delta_{s,g_j} &= \omega_{rs}\sum_i\theta_i\delta_{r,g_i}\sum_j\theta_j\delta_{s,g_j} \\ &= \omega_{rs}n_rn_s, \end{aligned} \tag{5}$$

where we have made use of (1) again. But the total number of places one can place an edge between groups r and s is n_rn_s, which means that the average probability of an edge is simply ω_{rs}.

As mentioned above, most of the networks we look at in practice are very sparse—the average probability of an edge is much less than one. For this reason a uniform prior on ω_{rs} is not appropriate. Rather, we need a prior

that favors values of ω_{rs} in the vicinity of the average probability of an edge in the network as a whole, which is $p = 2m/n^2$, where m is the total number of edges in the network. Here, as previously [10], we use a maximum-entropy prior conditioned on fixing the expected value of ω_{rs} to be equal to p, which yields the exponential distribution $P(\omega) = (1/p)e^{-\omega/p}$.

With these choices of priors on θ and ω, the integrals in Eq. (4) can be completed and we get

$$P(A|g,k) = \prod_r n_r^{\kappa_r} \frac{(n_r-1)!}{(n_r+\kappa_r-1)!} \times \prod_{r<s} \frac{m_{rs}!}{(pn_rn_s+1)^{m_{rs}+1}} \prod_r \frac{m_{rr}!}{(\frac{1}{2}pn_r^2+1)^{m_{rr}+1}}, \tag{6}$$

where

$$\kappa_r = \sum_i d_i \delta_{r,g_i} \tag{7}$$

is the sum of the degrees of the nodes in group r, and we have again discarded an overall multiplying constant.

III. PRIOR ON GROUP ASSIGNMENTS

Our goal is to use this model as the basis for a Bayesian model selection procedure to estimate the correct value of k for a given network. To do this, we need to specify a prior on the group assignments, meaning a joint probability distribution $P(g,k)$ on the group labels and the number of groups. This then allows us to write

$$P(g,k|A) = \frac{P(g,k)P(A|g,k)}{P(A)}, \tag{8}$$

where $P(A|g,k)$ is given by Eq. (6). Given this distribution, we can either sum over k to get the posterior distribution on g, which allows us to do community detection, or sum over g to get the posterior distribution on k, which allows us to choose a value for k. It is the latter computation that is our primary focus in this paper. In practice, we cannot perform the sum over g exactly, but we can approximate it using Markov chain Monte Carlo.

As is often the case with Bayesian methods, the tricky part of the calculation (or one of the tricky parts) is choosing the prior $P(g,k)$. It turns out that in the present case this choice can have a substantial impact on the results, making the difference between a method that works well in most cases and a method that does not. Moreover, some obvious choices of prior, including ones in common use elsewhere in the literature, result in methods that work poorly, so it is not a matter of simply grabbing a well-understood prior off the shelf.

A. Dirichlet prior

Suppose for the moment that the number of groups k is known and let us then ask what the prior $P(g|k)$ on group assignments should be. (This is not a very realistic assumption, since our whole purpose here is to estimate k, but the exercise is nonetheless instructive as we will see.) Our first guess at $P(g|k)$ might be to choose a flat probability distribution: all assignments g are equally likely, or equivalently every node is equally likely to be in each of the k groups. This approach is used, for example, in [15], but, as pointed out by Peixoto [12], it is unlikely to give good results. If nodes are distributed with equal probability among the groups then when n is large (as it usually is) the sizes of the groups will be sharply peaked around n/k. Any state with group sizes significantly different from n/k will occur rarely. This is strongly at odds with the observed situation in real-world networks, where we commonly encounter heterogeneous group sizes.

An alternative approach, therefore, and the one that is most commonly adopted in the literature, is to assume a uniform distribution not over assignments g but over the sizes of the groups n_r. That is, all possible sets of sizes are equally likely, subject only to the constraint that they sum to the size of the whole network: $\sum_r n_r = n$. The standard way to achieve this is to specify the expected fraction $\gamma_r \in [0,1]$ of the network occupied by each group r, such that $\sum_r \gamma_r = 1$, then assign nodes to groups independently at random, with probability γ_r of being assigned to group r. If the γ_r are themselves drawn from a uniform distribution, this makes the distribution of the sizes of the groups uniform. (It is obvious that this makes the *expected* sizes of the groups uniform, since the expected sizes are proportional to γ_r, but it is not entirely obvious that the sizes themselves are also uniform. However, the results given in Section III B provide a proof that they are.)

The probability of generating a particular group assignment g using this process is

$$P(g|\gamma,k) = \prod_{i=1}^n \gamma_{g_i} = \prod_{r=1}^k \gamma_r^{n_r}. \tag{9}$$

Since $\sum_r \gamma_r = 1$, the points defined by the values γ_r fall on a regular $(k-1)$-dimensional simplex, which has volume $1/(k-1)!$. So a uniform distribution over values of γ has probability density $P(\gamma|k) = (k-1)!$ and integrating (9) over the simplex then gives [8–10]

$$P(g|k) = \int P(g|\gamma,k)P(\gamma|k)\,\mathrm{d}\gamma = \frac{(k-1)!}{(n+k-1)!}\prod_{r=1}^k n_r! \tag{10}$$

The uniform distribution over values of γ is a special case of the so-called *Dirichlet prior*, a general prior on a simplex that includes this choice but also includes a spectrum of non-uniform choices as well.

B. Non-parametric prior

The derivation in the previous section is a standard one, but in a sense it is needlessly complicated. If our goal is simply to choose group sizes such that all choices are equally likely then assign nodes to those groups at random, why not just do so directly, without introducing other parameters? There are $\binom{n+k-1}{k-1}$ possible choices of k groups such that their sizes sum to n. Let us choose uniformly among these, then the number of ways of placing the nodes in the groups is given by the multinomial coefficient $n!/\prod_r n_r!$, so the probability of any given assignment g of nodes to groups is

$$P(g|k) = \frac{1}{\binom{n+k-1}{k-1}\, n!/\prod_r n_r!} = \frac{(k-1)!}{(n+k-1)!} \prod_{r=1}^{k} n_r! \tag{11}$$

which recovers Eq. (10) without the need for the parameters γ_r.

(A corollary of this result is that the Dirichlet process of Section III A does indeed generate a uniform distribution over possible group sizes, as claimed, since the distributions (10) and (11) are identical.)

C. Non-empty groups

A possible objection to these methods for generating assignments g (widely used though they are) is that they allow groups to be empty. It is unclear what the meaning is of an empty group. If someone were to hand you a network with two clear groups in it, then tell you that really there are three groups but one of them is empty, you might justifiably say that this is not a meaningful statement.

One advantage of the non-parametric formulation of Section III B is that generalizes easily to the case where groups are required to be non-empty. One need simply replace the binomial coefficient for the number of ways of generating the group sizes with the corresponding coefficient for non-empty groups, which is $\binom{n-1}{k-1}$. Then

$$P(g|k) = \frac{1}{\binom{n-1}{k-1}\, n!/\prod_r n_r!}. \tag{12}$$

D. Choice of the number of groups

We turn now to the choice of prior $P(k)$ on the number of groups itself. Again one's first guess at a prior might be a flat distribution with all choices equally likely. For non-empty groups as in Section III C, the possible choices for number of groups range from 1 to n, so a flat prior would have $P(k) = 1/n$ in this range and zero for all other values of k. This choice has been made in some previous work [9], but we find it to give poor results, placing too much weight on high values of k and significantly overestimating the number of groups in well-understood test cases. In practice one must use a strongly decreasing prior on k to achieve consistent results. Previous authors have given qualitative arguments in favor of a prior going as $1/k!$ [8] or even steeper [16].

In this paper we take a somewhat different approach and do away with an explicit prior on k, instead employing a generative process for group assignments g that automatically incorporates the choice of the number of groups in a simple way. The process we use, a queueing-type mechanism which is a variant on the "restaurant" processes of traditional probability theory, is as follows. Take the n nodes in random order and place the first one in group 1. Then for each subsequent node either (a) with probability $1-q$ place it in the same group as the previous node or (b) with probability q make it the first node in the next group. Note that this process never generates an empty group. All groups contain at least one node.

The number of possible orders of the nodes in this process is $n!$, with each one occurring with equal probability $1/n!$. If the process generates k groups in total then there must be $k-1$ new groups started and, since every node except the first has equal chance q of starting a new group, the probability of generating k groups with sizes $n_1 \ldots n_k$ is

$$(1-q)^{n_1-1} q (1-q)^{n_2-1} q \ldots q (1-q)^{n_k-1} = q^{k-1}(1-q)^{n-k}, \tag{13}$$

where we have made use of the fact that $\sum_r n_r = n$. Furthermore, there are $\prod_r n_r!$ ways of rearranging the nodes within each group that give rise to the same assignment g. Hence the probability of generating any given assignment under our proposed process is

$$P(g,k) = \frac{1}{n!}\, q^{k-1}(1-q)^{n-k} \prod_{r=1}^{k} n_r! \tag{14}$$

Given that $P(g,k) = P(k)P(g|k)$ and comparing with Eq. (12), we see that this process is equivalent to the process of Section III C if one chooses a prior $P(k)$ on the number of groups thus:

$$P(k) = \frac{P(g,k)}{P(g|k)} = \binom{n-1}{k-1} q^{k-1}(1-q)^{n-k}, \tag{15}$$

with $1 \le k \le n$. In other words, the number of new groups $k-1$ created in the generating process has a binomial distribution (as one can easily derive by considering the process directly).

For our purposes it will be convenient to parametrize the probability q by $q = \mu/(n-1)$, so that μ is the expected number of new groups started during the assignment process, which is one less than the total number of groups. Then

$$P(g,k) = \frac{(1-q)^n}{qn!} \frac{\mu^k}{(n-\mu-1)^k} \prod_{r=1}^{k} n_r! \tag{16}$$

The leading factor of $(1-q)^n/qn!$ is independent of both g and k and will cancel out of subsequent calculations.

From III: we have the P(g,k) prior (first derived p(g|k) using stars & bars, then CRT for P(k)) → have P(g,k|A)

E. Choice of parameter value

Our prior on g, k now has just one parameter μ, whose value we have yet to choose. One way to proceed is to take the Bayesian approach a step further and place a prior on the prior—a so-called hyperprior, meaning a probability distribution on μ. In principle one could go even further and place a hyperhyperprior on the hyperprior too, and so forth *ad infinitum*. This process usually yields diminishing returns, however, and one normally stops at some point, simply fixing a value for the parameters. In the present case, we choose to halt the process at the level of the parameter μ. In our tests we have found that a value $\mu = 1$ works well, so that, neglecting constants

$$P(g,k) = (n-2)^{-k} \prod_{r=1}^{k} n_r! \tag{17}$$

although other values around 1 give basically the same results, so the method does not seem sensitive to the precise choice we make. No doubt a hyperprior centered roughly around 1, such as a suitably sized normal distribution, would also give similar results, but we don't see any advantage to taking this approach.

It is interesting to note that when $q = \mu/(n-1)$ with $\mu = 1$, and taking the limit of large n, the prior on k, Eq. (15), becomes

$$P(k) = \frac{e^{-1}}{(k-1)!}. \tag{18}$$

In other words, this choice is essentially equivalent to the $1/k!$ prior proposed previously on heuristic grounds [8].

$\sum_g P(k,g|A) = P(k|A)$ & but sum is hard to calc so we use MCMC instead

IV. MONTE CARLO ALGORITHM

Given the prior, Eq. (17), on g, k, we can now write down the complete posterior distribution (8) on the same quantities. Then by summing over all values of g we can find the probability distribution $P(k|A)$ and hence deduce the most likely value of k. Unfortunately the sum over g is hard to do: it has k^n terms, which is a very large number in most cases, making exhaustive numerical evaluation impossible, and no simple scheme presents itself for performing the sum analytically. Instead therefore we estimate the distribution over k by Markov chain Monte Carlo sampling.

The Monte Carlo scheme we propose employs steps of two types:

- **Type 1:** Moving a single node from group to group. This type of move includes processes that decrease the number of groups (if the node moved is the last of its group) and processes that keep the number of groups constant (if the node moved is not the last of its group).
- **Type 2:** Moving a single node to a newly created group, thereby increasing the value of k by one.

A sufficient condition for a correct Monte Carlo algorithm is that the algorithm satisfy the requirements of *ergodicity* and *detailed balance* [17]. The requirement of ergodicity says that every state of the system must be accessible from every other by a finite sequence of Monte Carlo steps. It is trivial to show that this condition can be satisfied by steps of the kind described above that move individual nodes from one group to another.

More demanding is the requirement of detailed balance, which in the present situation says that the rate $R(g,k \to g',k')$ to go from a state (g,k) to another state (g',k') and the rate $R(g',k' \to g,k)$ to go back again must satisfy

$$\frac{R(g,k\to g',k')}{R(g',k'\to g,k)} = \frac{P(g',k'|A)}{P(g,k|A)} = \frac{P(g',k')}{P(g,k)} \times \frac{P(A|g',k')}{P(A|g,k)}, \tag{19}$$

where we have used Eq. (8). From Eq. (17) we have

$$\frac{P(g',k')}{P(g,k)} = (n-2)^{k-k'} \frac{\prod_{r=1}^{k'} n_r'!}{\prod_{r=1}^{k} n_r!}, \tag{20}$$

where n_r' are the sizes of the groups for group assignment g'.

We use a traditional accept/reject Monte Carlo scheme in which we repeatedly propose a potential move then either accept or reject that move with probabilities chosen to satisfy the detailed balance condition. Thus the rate $R(g,k \to g',k')$ divides into the product of the probability π of proposing the move in question and the probability α of accepting it: $R(g,k \to g',k') = \pi(g,k \to g',k')\,\alpha(g,k \to g',k')$. Then

$$\frac{R(g,k\to g',k')}{R(g',k'\to g,k)} = \frac{\pi(g,k\to g',k')}{\pi(g',k'\to g,k)} \times \frac{\alpha(g,k\to g',k')}{\alpha(g',k'\to g,k)}. \tag{21}$$

The algorithm we propose is as follows:

1. (a) On each step of the algorithm, with probability $1-1/(n-1)$ we propose a move of type 1. Specifically, if $k = 1$ we do nothing (because there are no possible moves that move a node from one group to another). Otherwise, when $k > 1$, we choose a pair of distinct group labels r, s uniformly at random from the set of all such pairs in the range $1 \ldots k$, then choose a single node uniformly at random from group r and move it to group s.

   (b) If, in the process, we remove the last remaining node from group r, leaving that group empty, we relabel the non-empty groups so that their labels run from $1 \ldots k-1$, and we decrease k by one. In practice, the most efficient way to do the relabeling is just to change the current group k to have label r (unless $r = k$, in which case no relabeling is necessary).

2. (a) Otherwise, with probability $1/(n-1)$ we propose a move of type 2. Specifically, we choose a pair of distinct group labels r, s uniformly at random from the set of all such pairs in the range $1 \ldots k+1$, relabel group r as group $k+1$, and create a new empty group r (unless $r = k+1$, in which case we simply create a new empty group $k+1$ and no relabeling is necessary). Then we choose a node uniformly at random from group s and move it to the newly created group r.

   (b) If, in the process, we remove the last remaining node from group s, leaving that group empty, we change group $k+1$ to have label s. Otherwise we increase k by 1. (Note that k can never become greater than n during this process, since doing so would always involve removing the last node from group s, which precludes increasing k any further.)

3. Once we have proposed our move from state (g, k) to state (g', k'), we accept it with acceptance probability

$$\alpha(g, k \to g', k') = \min\left(1, \frac{P(A|g', k')}{P(A|g, k)}\right). \tag{22}$$

   If the move is accepted, g', k' becomes the new state of the system. Otherwise the system remains in the old state g, k. Note, crucially, that the probabilities appearing on the right-hand side of (22) are of the form $P(A|g, k)$, and not $P(g, k|A)$ as you might expect if you are familiar with standard (Metropolis–Hastings) Monte Carlo methods.

4. Repeat from step 1.

To see that that this algorithm does indeed satisfy the condition of detailed balance, Eq. (19), consider first a move of type 1 that moves a node from group r to group s, where $n_r > 1$ so that the node moved is not the last node in group r and the value of k does not change. Thus $k' = k$ and Eq. (20) simplifies to

$$\frac{P(g', k')}{P(g, k)} = \prod_{t=1}^{k} \frac{n_t'!}{n_t!} = \frac{n_s'}{n_r}, \tag{23}$$

with the terms for all groups other than r and s canceling. Moves of type 1 are performed with probability $1 - 1/(n-1)$, there are $k(k-1)$ possible choices of distinct groups r, s, and n_r nodes to choose from in group r, so the total probability of proposing a specific move of a specific node is

$$\begin{aligned}\pi(g, k \to g', k') &= \left(1 - \frac{1}{n-1}\right) \times \frac{1}{k(k-1)} \times \frac{1}{n_r} \\ &= \frac{n-2}{(n-1)k(k-1)n_r}.\end{aligned} \tag{24}$$

Similarly, the probability of proposing the reverse move, in which the same node is moved from group s to group r, is

$$\pi(g', k' \to g, k) = \frac{n-2}{(n-1)k(k-1)n_s'}. \tag{25}$$

And the ratio of the two is

$$\begin{aligned}\frac{\pi(g, k \to g', k')}{\pi(g', k' \to g, k)} &= \frac{n-2}{(n-1)k(k-1)n_r} \times \frac{(n-1)k(k-1)n_s'}{n-2} \\ &= \frac{n_s'}{n_r},\end{aligned} \tag{26}$$

which is precisely equal to Eq. (23).

We can also demonstrate an equivalent result for moves that change the value of k. Consider a move of type 1 that removes the last node from group r and moves it to group s, thereby reducing the number of groups by 1, so that $k' = k - 1$. For such a move, Eq. (20) becomes

$$\frac{P(g', k')}{P(g, k)} = (n-2)n_s'. \tag{27}$$

The probability of proposing such a move is again given by Eq. (24), except that in this case $n_r = 1$, so the expression simplifies to

$$\pi(g, k \to g', k') = \frac{n-2}{(n-1)k(k-1)}. \tag{28}$$

The reverse of such a move is a move of type 2, in which a node in group s becomes the founding member of new group r. Moves of type 2 are performed with probability $1/(n-1)$, there are $(k'+1)k' = k(k-1)$ ways of choosing the labels r, s, and n_s' ways of choosing the node to be moved. Thus the total probability of proposing such a move is

$$\pi(g', k' \to g, k) = \frac{1}{(n-1)k(k-1)n_s'}. \tag{29}$$

Taking the ratio of (28) and (29), we get

$$\begin{aligned}\frac{\pi(g, k \to g', k')}{\pi(g', k' \to g, k)} &= \frac{n-2}{(n-1)k(k-1)} \times (n-1)k(k-1)n_s' \\ &= (n-2)n_s',\end{aligned} \tag{30}$$

which is equal to (27).

Finally, there is one further class of moves that have to be considered separately, namely moves of type 2 that remove the last node from group s and make it the initial node in a new group r. By contrast with other moves of type 2, these moves do not change the value of k. Moreover they don't change the product $\prod_r n_r!$ either, so Eq. (20) is simply

$$\frac{P(g', k')}{P(g, k)} = 1. \tag{31}$$

At the same time, the probability of proposing such a move is the same in both directions, equal to $1/[(n-$

Didit look closely at detailed balance proof, but mechanics seem straitforward.

$1)k(k+1)]$, and hence the ratio of the forward and backward proposal probabilities is 1, which is equal to (31).

Thus for all moves of all types we have

$$\frac{\pi(g,k\to g',k')}{\pi(g',k'\to g,k)} = \frac{P(g',k')}{P(g,k)}. \tag{32}$$

Equating Eqs. (19) and (21) and making use of (32), we then find that the detailed balance condition becomes

$$\frac{\alpha(g,k\to g',k')}{\alpha(g',k'\to g,k)} = \frac{P(A|g',k')}{P(A|g,k)}. \tag{33}$$

If we can choose the acceptance ratios to satisfy this relation, then Eq. (19) will be obeyed. But the choice in Eq. (22) trivially satisfies (33), hence detailed balance is obeyed and the proposed algorithm will sample correctly from the distribution $P(g,k|A)$.

A. Implementation

Implementation is a relatively straightforward translation of the algorithm described above into computer code. We maintain not only a record of the group assignment g_i of each node but also a separate, unordered list of the members of each group, which allows us to choose a random member of a group rapidly, and to efficiently relabel all members of a group when required. We calculate the logarithm of the ratio $P(A|g',k')/P(A|g,k)$, rather than the ratio itself, to avoid numerical problems with powers and factorials (which can become large) and only take the exponential at the end to determine the acceptance ratio, Eq. (22). We also employ a look-up table of log-factorials to speed their calculation, and a running record of the n_r and m_{rs}, updated after every accepted move, so as to avoid recalculating these values repeatedly.

As a practical matter, the relabeling process for moves of type 2 can be slow if group r contains many nodes, so in our implementation we always give new groups label $k+1$, which frees us from having to relabel any nodes other than the one node that is placed in the newly created group. Technically, this means that our Monte Carlo algorithm does not sample labelings g with the true probability of Eq. (17): group k will typically be the newest group and therefore smaller on average than the others. However, the algorithm does still sample *divisions* of the network into groups and values of k with the correct probability, and these are the only physical quantities we care about. The group labels themselves have no meaning—they exist only as a mathematical convenience for the purposes of notation. The only meaningful quantities are the value of k and the division into groups, and these are correctly generated. (And if one were concerned to sample labelings g correctly, one could do so easily by taking the labelings generated by the algorithm and randomly permuting the labels.)

The initial assignment of nodes to groups is drawn at random from the prior distribution defined in Section III D. In order to avoid any bias in the results, instead of just setting $\mu = 1$ we use a value of μ that is itself chosen randomly. In the example calculations presented in Section V, the value of μ for the initial assignment is chosen uniformly in the interval from 0 to 100, meaning in practice the initial number of groups lies approximately uniformly in this range. (In the actual Monte Carlo calculation, however, we always use $\mu = 1$, as described in Section III E.)

Our code is implemented in C, and performs about a million Monte Carlo steps per second on a typical desktop computer (circa 2017), which puts the analysis of large networks, up to hundreds of thousands of nodes or more, within reach. Our code is available for download on the web—see Ref. [18].

B. Relation to previous approaches

In a previous paper [10] we proposed a slightly different algorithm for determining the number of communities, based on the same principles used here but different in detail. We expect the present algorithm to be more efficient than the earlier one, primarily because of the way it incorporates the prior on group assignments $P(g,k)$ into the proposal probability π rather than the acceptance probability α. However, we also believe the derivation given here is more appropriate than that given in the previous paper, particularly in its focus on the prior $P(g,k)$ on group assignments.

The argument given in the previous paper differs from that given here in two ways. First, in the previous paper we advocated using a flat prior $P(k)$ on the number of groups, whereas in this paper we argue for a decreasing prior going as $1/k!$. On the other hand, the probability $P(g|k)$ given in the previous paper omits a factor of $k!$, equal to the number of ways the labels on a given partition of the network can be permuted without changing the partition. These two factors of $k!$ cancel, leaving the equations essentially unchanged. Thus the formulas and algorithm given in the previous paper are essentially equivalent to those given here, but the motivation differs, with the arguments given here being, in our opinion, the correct ones.

V. EXAMPLE APPLICATIONS

In this section we apply our method to a wide range of networks and find it to give good results in most cases. To evaluate performance under controlled conditions, we test the method on several large sets of computer-generated networks, created using both the stochastic block model and the widely used LFR benchmark. To test the method under real-world conditions we have also applied it to a range of observed networks, including a

IV: Use MCMC to sample from (g,K) to get an optimal pair. Didn't look closely at details, but the idea is clear.

number of staples of the community detection canon, as well as a large example network with over 300 000 nodes.

A. Computer-generated networks

In order to explore the performance of our method systematically we have tested it on a range of computer-generate ("synthetic") networks. These are networks with known community structure planted within them, generated using random graph models. Using synthetic networks allows us to vary the number of planted communities and quantify the extent to which our algorithm is able to correctly recover that number.

In our first set of tests, we use networks generated using the standard (non-degree-corrected) stochastic block model [13, 14]. Figures 1a and 1b show results for two different sets of networks. Each panel shows the number of communities k inferred by our method plotted against the known number planted in the network as the latter is varied (blue circles in the plots). In Fig. 1a the size of the network is held fixed as the number of communities in increased, while in Fig. 1b the size of the communities is held fixed, so that the size of the network increases with the number of communities. As the figures show, in both cases the algorithm infers the correct number of communities with high accuracy for values of k up to about 20.

For higher values it has a tendency to underestimate the number of communities. These calculations, however, are for runs of the Monte Carlo algorithm that start with a random assignment of nodes to groups, as described in Section IV A. Also shown in the figures are the results of runs on the same networks in which the Monte Carlo algorithm was started with group assignments corresponding exactly to the planted ground-truth community division (red triangles). As the figures show, for this choice of initialization the algorithm finds the correct number of communities for the entire range of values of k explored. These results suggest that the underestimation of k arises not because the correct group assignment fails to maximize the posterior probability $P(k|A)$, but rather because the Monte Carlo algorithm has not run for long enough to find the maximum. That is, the method is theoretically sound but the numerical calculation becomes too demanding as k becomes large. Possibly this problem could be solved with a more efficient Monte Carlo sampling scheme, although it seems likely that some similar issue will eventually arise no matter what sampling scheme is used. The fundamental problem is that the number of possible group assignments k^n increases very rapidly with k, so it becomes unfeasible to explore the space of assignments effectively when k is very large. On the other hand, since, as Figs. 1a and 1b show, the method only underestimates the value of k and does not overestimate, the algorithm gives a lower bound on the number of communities in the network, which may well have some utility even when the exact value of k is not found. Note that the fact that the algorithm underestimates k does not appear to be a result of the initial conditions. As described in Section IV A, the algorithm starts with an initial number of groups anywhere up to 100, so the initial conditions are at least as likely to overestimate k as underestimate. It seems likely, therefore, that the underestimates we see in the final results are a consequence of the Monte Carlo sampling method and not of the initial conditions.

The examples in Fig. 1a and 1b assume so-called assortative network structure, meaning that there are more in-group edges than between-group edges in the network. Our method, however, should in principle be just as good at finding the number of communities in disassortative cases, where there are more between-group edges, or in mixed assortative/disassortative cases. Figure 1c shows results from tests on networks of mixed type, generated again using a stochastic block model but now taking a diagonally dominant matrix of edge probabilities and permuting the rows and columns to move some of the large matrix entries off the diagonal. As the figure shows, the algorithm does indeed perform well in this case, indeed it appears to perform better than in the purely assortative case of Figs. 1a and 1b. A possible explanation is that in the purely assortative case one can (erroneously) join together groups and produce another assortative network with strong community structure, but in a disassortative or mixed case joining groups is not guaranteed to produce another network with strong structure.

The parameter values used in each of these examples mean that the networks generated have quite prominent community structure—the number of communities varies from network to network but all have strong structure that should be relatively straightforward to detect. It is interesting to ask how the method fares if we make the structure weaker. Figure 2 shows the results of a set of tests on networks with varying difference between the number of in-group and between-group connections. The horizontal axis is normalized to place the so-called detectability threshold at ± 1. This is the point at which communities become formally undetectable in the network because the structure is too weak [20–22]. As the figure shows, our method gets the value of k correct virtually all of the time outside of the undetectable region (marked by the vertical dashed lines in the figure), except again for assortative networks with very large numbers of communities (such as the right-hand portion of the curve for $k = 32$).

Widely used though it is, one could argue that the stochastic block model is not a very realistic model. The networks it generates have Poisson degree distributions within each community, for instance, and in the cases studied here we have also limited ourselves to communities of uniform size. An alternative model that avoids these limitations is the LFR benchmark model of Lancichinetti, Fortunato, and Radicchi [19]. This model is essentially a special case of the degree-corrected stochastic block model of Section II, with both the degrees and

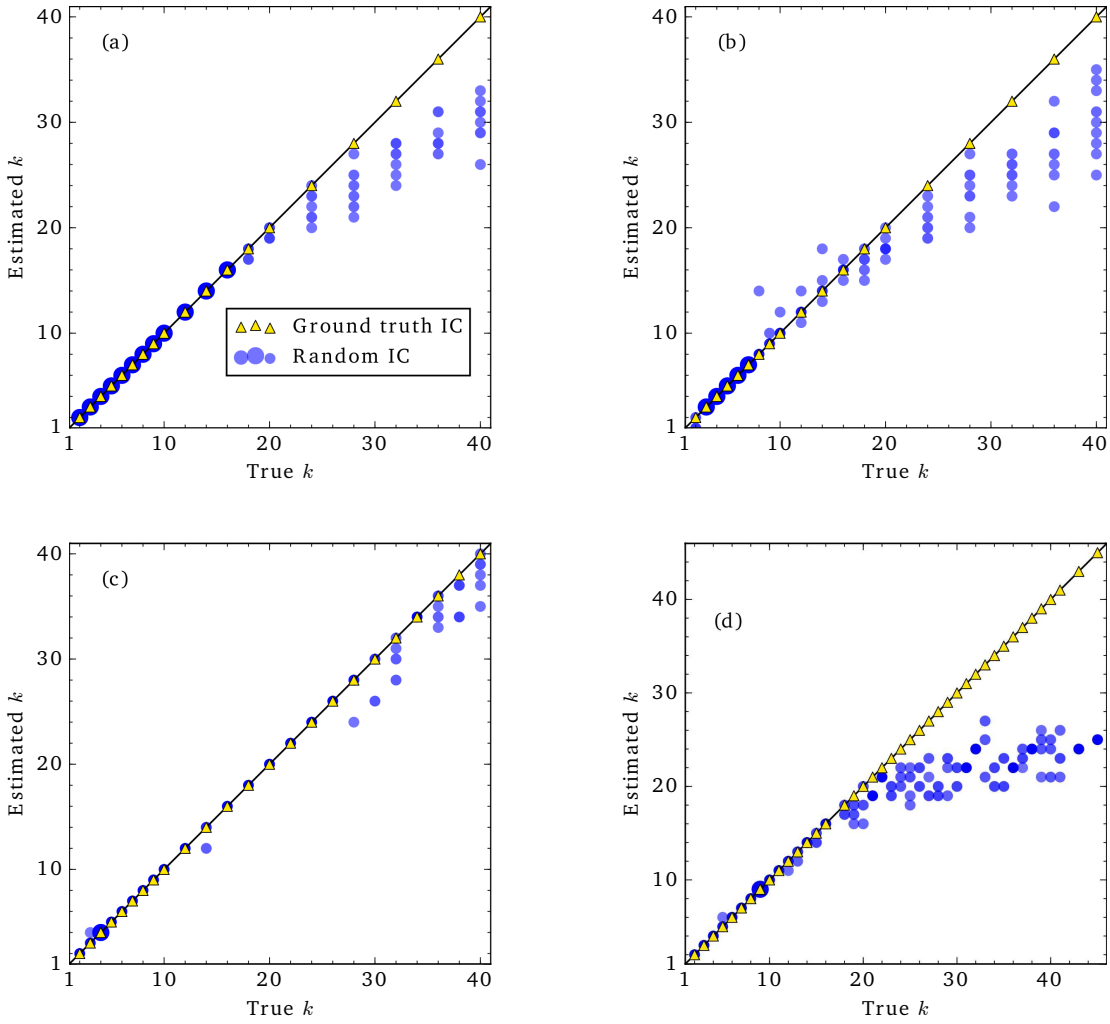

FIG. 1: Tests of the method on synthetic networks. In each panel, circles represent results derived from Monte Carlo runs with random initial assignment of nodes to group ("random initial conditions"), while triangles represent runs started with the known correct assignments ("ground truth initial conditions"). (a) Networks generated using the stochastic block model with $n = 1000$ nodes, mean degree 30, and equally sized groups with 90% of connections with groups and 10% between groups. (b) Networks generated using the stochastic block model with groups of fixed size 250 nodes (so that network size varies with the number of groups k), and each node having an average of 16 in-group connections and 8 out-group connections. (c) Networks generated using the same stochastic block model as in (a) but with the rows and columns of the matrix of edge probabilities permuted to produce a mixed assortative/disassortative structure. (d) Networks generated using the LFR benchmark model of [19], which is parametrized by it maximum and minimum group sizes. In these tests we used minimum groups sizes between 10 and 80 nodes, and maximum group size equal to five times the minimum. Ten Monte Carlo runs were performed for each network of 2000 steps per node each, with the distribution over k being calculated from the final 1000 only.

the sizes of the communities drawn from power-law distributions, giving the networks similar features to those seen in many real-world examples.

Figure 1d shows the results of tests of our method on LFR networks generated with the same model parameters as those used by Lancichinetti and Fortunato [23] in a widely cited study. The results are similar to those for the stochastic block model: the method performs well for smaller values of the number of groups k but tends to underestimate as the value of k gets larger. If, however, the Monte Carlo algorithm starts with an initial group assignment equal to the planted structure, then it reliably finds the correct number of groups for all values of k, again suggesting that the problem is in the time available for equilibration and sampling, rather than any fundamental issue with the approach. If one were able to sample from the entire distribution $P(g, k|A)$ in reasonable time, one should find the correct number of groups.

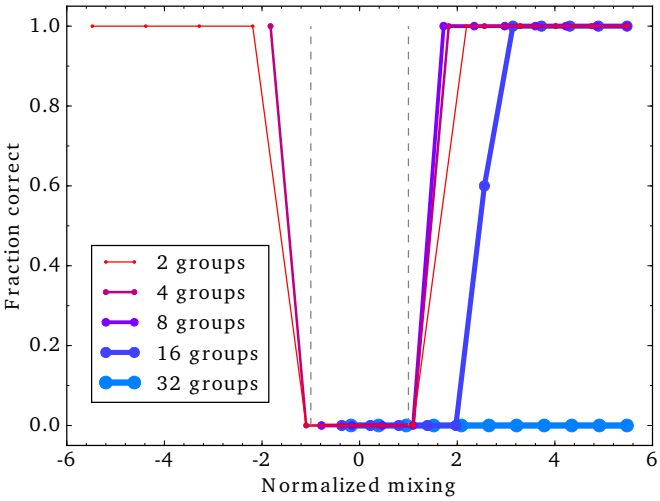

FIG. 2: Fraction of runs on which the algorithm correctly estimates the number of groups k in tests on networks generated using the stochastic block model, as a function of the normalized mixing parameter $(c_{\rm in} - c_{\rm out})/\sqrt{c}$. Networks have $n = 1000$ nodes, average degree $c = 30$ and $k = 2$, 4, 8, 16, 32, with mixing varying from perfect assortativity (right-hand size of the plot) to perfect disassortativity (left-hand side). The dashed gray lines at ± 1 denote the theoretical detectability thresholds. Fifty Monte Carlo runs were performed for each network of 2000 steps per node each, with the distribution over k calculate from the final 1000 only.

B. Real-world networks

As a complement to the synthetic tests of the previous section, as have also tested our method on a range of real-world networks. There exist a number of well studied example networks in the literature that have widely agreed upon ground-truth community divisions, based in part on knowledge of the specific systems the networks describe and in part on consensus derived from repeated analyses with many different community detection algorithms. Figure 3 shows results for four such networks.

The first column in Fig. 3 shows results for tests on the "karate club" network of Zachary [24], perhaps the best known and most widely used benchmark of community detection. This small social network is universally agreed to contain two clear communities, and when applied to the network our method firmly favors $k = 2$. The top panel in the figure shows a histogram of the values of k sampled by the Monte Carlo algorithm on this network, and the probability shows a clear peak for two communities.

It could be argued, however, that looking directly at the values of k generated by the Monte Carlo algorithm has the potential to be misleading in some cases. Imagine, for instance, that a network breaks apart into two large groups that occupy most of the network, plus a third group with only a very few nodes in it. In this situation one might be justified in saying that the network really only contains two groups, not three. One can capture this kind of situation by defining an effective number of groups

$$k_{\rm eff} = e^S, \qquad S = -\sum_{r=1}^{k} \frac{n_r}{n} \log \frac{n_r}{n}. \tag{34}$$

Here S is the entropy of the group assignment, which has a maximum value of $\log k$ when the groups are equally sized, so that $k_{\rm eff} = e^S = k$ in this case. On the other hand, if there are a few small groups in the network and the remainder are large and equally sized then the measure will ignore the small groups to a great extent and $k_{\rm eff}$ will be roughly equal to the number of large groups only.

The second panel in column 1 of Fig. 3 shows results for this alternative measure of group number, as calculated using our Monte Carlo algorithm on the karate club network. It is straightforward to show that $k_{\rm eff} \le k$ strictly, so by necessity the distribution in the second panel is to the left of that in the top panel. There are three clear peaks visible in the distribution of $k_{\rm eff}$, corresponding to states with one, two, and three groups, so it seems reasonable to assume that these are "real" divisions of the network. The peak for $k_{\rm eff} = 1$ is the highest, but the area under the peak for $k_{\rm eff} = 2$ is greater, so two groups, which is the widely accepted number, still seems to be preferred overall.

The third panel in column 1 shows the group assignment g with the maximum likelihood sampled by the Monte Carlo algorithm, which in this case corresponds closely to the accepted community structure of the karate club network. The fourth panel shows an alternative visualization of the community structure, a plot of the adjacency matrix of the network where the rows and columns have been ordered so as to put the two groups in contiguous blocks. As we can see, this places most edges within blocks and only a few between blocks, as we would expect for a network with strong community structure. Finally, in the bottom panel of the column, we show a visualization of the group structure itself, a "meta-network" in which the nodes represent the groups and edges represent connections between groups. In this simple case the meta-network does not offer much insight, since it consists of just two meta-nodes and a single edge, but in more complicated situations with larger numbers of groups—including some of the others in Fig. 3—it can be a useful tool.

The remaining three columns of Fig. 3 show corresponding analyses for three further networks: the American college football network studied in [1], the network of fictional character interactions in the novel *Les Misérables* by Victor Hugo [25], and a disassortative example, the network of word adjacencies of adjectives and nouns in the novel *David Copperfield* by Charles Dickens [26]. In each case the algorithm finds the accepted number of communities—six, eleven, and two, respectively. The distributions of $k_{\rm eff}$ largely agree with those

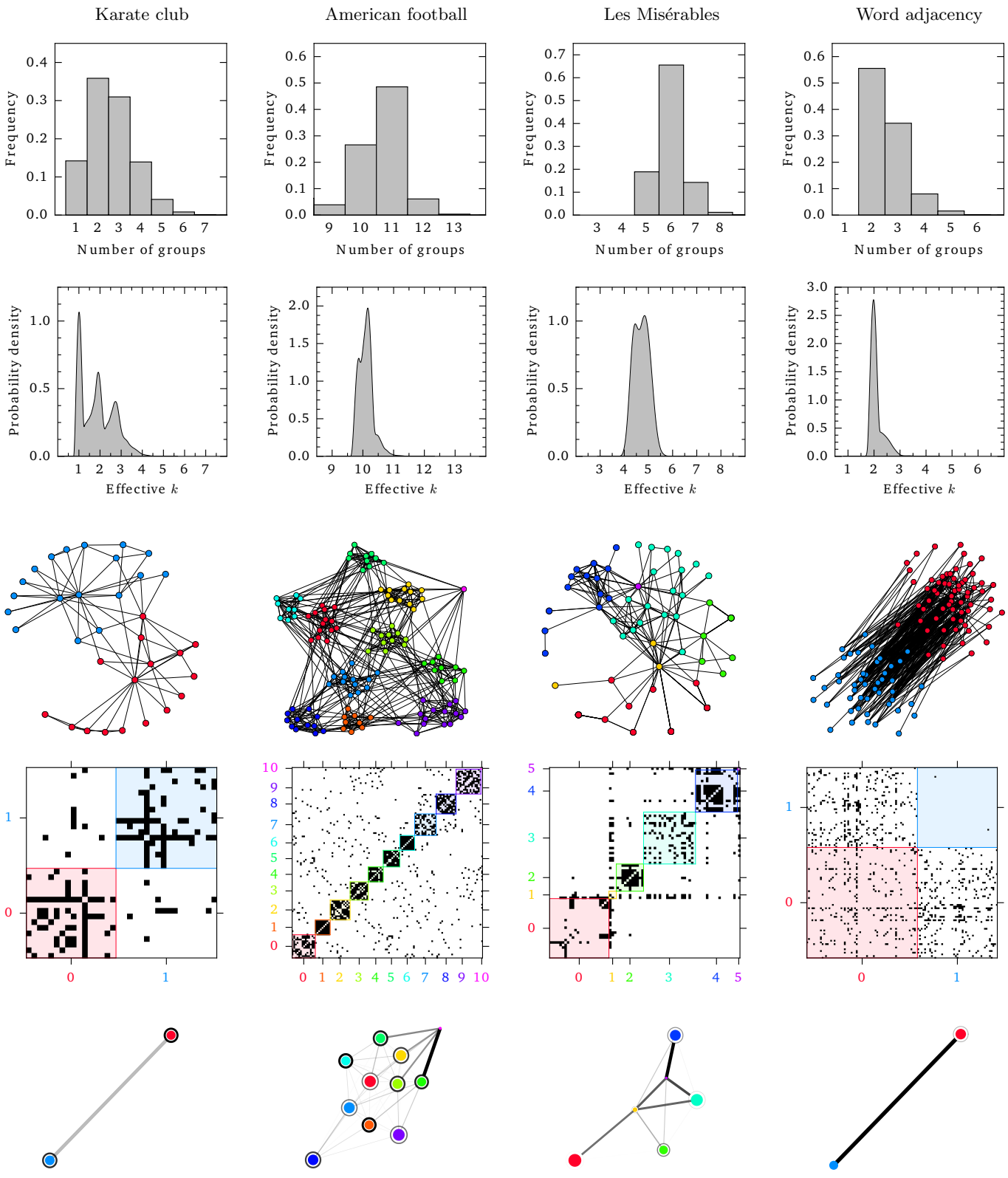

FIG. 3: Results for four real-world networks. Each column shows results for one network, as indicated. From top to bottom the results shown are the posterior probability distribution of the number of groups, the distribution of the effective number of groups calculated using the entropy measure of Eq. (34), the maximum likelihood community structure, the adjacency matrix, and the "meta-network" representation of the communities and their pattern of connectivity.

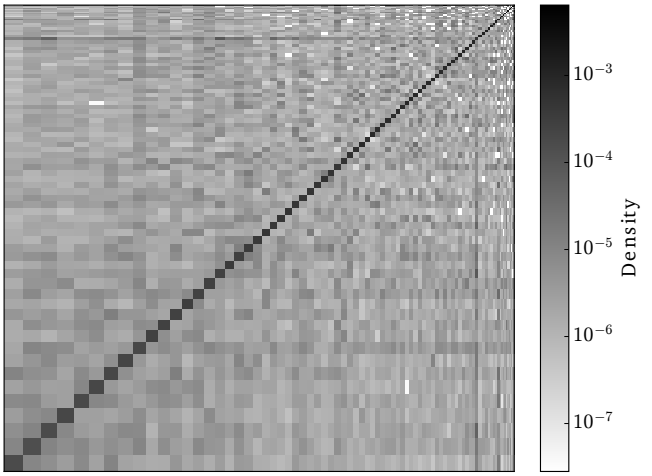

FIG. 4: A plot of the matrix $\omega_{g_i g_j}$ of connection parameters for a large network of copurchased products on Amazon.com. In this network nodes represents products and edges join pairs of products frequently purchased by the same buyer. In this calculation we performed ten runs of 10 000 Monte Carlo steps per node each and we display results from the run with the highest average likelihood during the last 1000. The calculation found 81 groups in total. Color is on a log scale for clarity.

for k, though for the *Les Misérables* and college football networks they favor five and ten groups respectively, one less than the peak in the distribution of k in each case, perhaps suggesting that these networks have one group that is small enough to be neglected.

These examples are all relatively small networks, up to around a hundred nodes in the larger cases, but our method is applicable in principle to much larger networks. As an example, Fig. 4 shows results for a network of e-commerce data, a copurchasing network of items sold by the online retailer Amazon.com [27]. In this network, which comes from the Stanford Large Network Dataset collection, the nodes represent 334 863 products for sale on the Amazon web site and edges between them indicate products that were frequently purchased by the same buyer. The figure shows a visualization of the inferred values of the edge probability parameters $\omega_{g_i g_j}$, again with the columns ordered so as to make the groups contiguous. As we can see, there appears to be strong assortative structure in the network, with the algorithm finding 81 groups in this case.

VI. CONCLUSIONS

In this paper we have described a method for determining the number of communities in a network with community structure. The method relies on a combination of Bayesian inference applied to the degree-corrected stochastic block model and a novel Monte Carlo algorithm. Much of the method's success turns on the appropriate choice of prior probability for the number of groups and we describe a variation on the "restaurant" processes of traditional model selection that appears to work well. We have illustrated the performance of the method with applications to a wide range of networks, including a diverse set of synthetic test networks and a number of real-world examples, one with over 300 000 nodes.

The primary limitation of the method as described is that the Monte Carlo algorithm appears not to equilibrate fully when the number of groups becomes large. A possible objective for future work, therefore, would be to find a method or algorithm that could sample the posterior distribution over group assignments faster, which would allow us to better apply the method to networks with large numbers of groups.

Acknowledgments

The authors thank Tiago Peixoto and Jia-Rong Xie for helpful conversations. This work was funded in part by the US National Science Foundation under grants DMS–1107796 and DMS–1407207 (MEJN), the UK Engineering and Physical Sciences Research Council under grant EP/K032402/1 (GR), the James S. McDonnell Foundation (MAR), the Simons Foundation (MEJN), and the Advanced Studies Centre at Keble College, Oxford (MEJN and GR).

[1] M. Girvan and M. E. J. Newman, Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* **99**, 7821–7826 (2002).

[2] S. Fortunato, Community detection in graphs. *Phys. Rep.* **486**, 75–174 (2010).

[3] S. Fortunato and D. Hric, Community detection in networks: A user guide. *Phys. Rep.* **659**, 1–44 (2016).

[4] M. S. Handcock, A. E. Raftery, and J. M. Tantrum, Model-based clustering for social networks. *J. R. Statist. Soc. A* **170**, 301–354 (2007).

[5] J. J. Daudin, F. Picard, and S. Robin, A mixture model for random graphs. *Statistical Computing* **18**, 173–183 (2008).

[6] P. Latouche, E. Birmelé, and C. Ambroise, Bayesian methods for graph clustering. In *Advances in Data Analysis, Data Handling, and Business Intelligence*, pp. 229–239, Springer, Berlin (2009).

[7] P. Latouche, E. Birmelé, and C. Ambroise, Variational Bayesian inference and complexity control for stochastic block models. *Statistical Modelling* **12**, 93–115 (2012).

[8] A. F. McDaid, T. B. Murphy, N. Friel, and N. Hurley, Improved Bayesian inference for the stochastic block model with application to large networks. *Computational Statistics and Data Analysis* **60**, 12–31 (2013).

[9] E. Côme and P. Latouche, Model selection and clustering in stochastic block models based on the exact integrated complete data likelihood. *Statistical Modelling* **15**, 564–589 (2015).

[10] M. E. J. Newman and G. Reinert, Estimating the number of communities in a network. *Phys. Rev. Lett.* **117**, 078301 (2016).

[11] X. Yan, Bayesian model selection of stochastic block models. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 323–328, Institute of Electrical and Electronics Engineers, New York (2016).

[12] T. P. Peixoto, Nonparametric Bayesian inference of the microcanonical stochastic block model. *Phys. Rev. E* **95**, 012317 (2017).

[13] B. Karrer and M. E. J. Newman, Stochastic blockmodels and community structure in networks. *Phys. Rev. E* **83**, 016107 (2011).

[14] P. W. Holland, K. B. Laskey, and S. Leinhardt, Stochastic blockmodels: Some first steps. *Social Networks* **5**, 109–137 (1983).

[15] R. Guimerà and M. Sales-Pardo, Missing and spurious interactions and the reconstruction of complex networks. *Proc. Natl. Acad. Sci. USA* **106**, 22073–22078 (2009).

[16] T. P. Peixoto, Hierarchical block structures and high-resolution model selection in large networks. *Phys. Rev. X* **4**, 011047 (2014).

[17] M. E. J. Newman and G. T. Barkema, *Monte Carlo Methods in Statistical Physics*. Oxford University Press, Oxford (1999).

[18] Computer code for the algorithm described in this paper is available for download from the world wide web at `www.umich.edu/~mejn/communities`.

[19] A. Lancichinetti, S. Fortunato, and F. Radicchi, Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **78**, 046110 (2008).

[20] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, Inference and phase transitions in the detection of modules in sparse networks. *Phys. Rev. Lett.* **107**, 065701 (2011).

[21] L. Massoulié, Community detection thresholds and the weak Ramanujan property. In *Proceedings of the 46th Annual ACM Symposium on the Theory of Computing*, pp. 694–703, Association of Computing Machinery, New York (2014).

[22] E. Mossel, J. Neeman, and A. Sly, Reconstruction and estimation in the planted partition model. *Probability Theory and Related Fields* **162**, 431–461 (2015).

[23] A. Lancichinetti and S. Fortunato, Community detection algorithms: A comparative analysis. *Phys. Rev. E* **80**, 056117 (2009).

[24] W. W. Zachary, An information flow model for conflict and fission in small groups. *Journal of Anthropological Research* **33**, 452–473 (1977).

[25] D. E. Knuth, *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, Reading, MA (1993).

[26] M. E. J. Newman, Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E* **74**, 036104 (2006).

[27] J. Yang and J. Leskovec, Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* **42**, 181–213 (2015).