Hierarchical core-periphery structure in networks

Austin Polanco[1] and M. E. J. Newman[1, 2]

[1]*Department of Physics, University of Michigan, Ann Arbor, Michigan 48109, USA*
[2]*Center for the Study of Complex Systems, University of Michigan, Ann Arbor, Michigan 48109, USA*

We study core-periphery structure in networks using inference methods based on a flexible network model that allows for traditional onion-like cores within cores, but also for hierarchical tree-like structures and more general non-nested types of structure. We propose an efficient Monte Carlo scheme for fitting the model to observed networks and report results for a selection of real-world data sets. Among other things, we observe an empirical distinction between networks showing traditional core-periphery structure with a dense core weakly connected to a sparse periphery, and an alternative structure in which the core is strongly connected both within itself and to the periphery. Networks vary in whether they are better represented by one type of structure or the other. We also observe structures that are a hybrid between core-periphery structure and community structure, in which networks have a set of non-overlapping cores that correspond roughly to communities, surrounded by a single undifferentiated periphery. Computer code implementing our methods is available.

I. INTRODUCTION

Networks are widely used as a compact and convenient mathematical representation of the connections between the elements of a complex system, such as data connections on the Internet, citations between papers, social contacts among people or animals, synaptic connections between brain cells, and biological and biochemical networks of many kinds [1, 2]. A significant amount of effort has been devoted in recent years to analyzing and understanding large-scale structure in such networks, especially community structure [3, 4], but also nested [3, 5] and overlapping [6, 7] communities and stratification [8, 9], as well as the related issues of embedding and graph representation learning [10]. In this paper we focus on a less well-studied form of large-scale structure, *core-periphery structure*, in which a network is divided into a densely connected core of nodes surrounded by a sparser periphery [11–13]. The observation of core-periphery structure communicates different information about a network from community structure. Where community structure deals with the identification of groups or types of nodes, core-periphery structure focuses on their roles and centrality. Core-periphery structure is integral to understanding the link between node position and function in networks, for instance in the Internet [14, 15], neuroscience [16], and economics [17].

A range of heuristic methods have been proposed in the past for detecting core-periphery structure in networks. In the simplest case the challenge is to take unlabelled network data and assign each node of the network in some automated fashion to either the core or the periphery. In more complex cases one may attempt to infer a onion-like sequence of deeper and deeper cores within cores.

The problem, however, is not entirely well posed, since we have not precisely defined what "core" and "periphery" mean. Various researchers have chosen to define them in various ways and there is, as a result, a corresponding spectrum of algorithmic approaches. Perhaps the oldest approach is the *k-core decomposition*, in which nodes are recursively removed from a network in order of increasing degree and the sequence in which they are removed defines a sliding scale from core to periphery [18]. This method essentially equates the core with high-degree nodes. Another well-established method is that of Borgatti and Everett [11], who defined a quality function akin to the well-known modularity function for community detection [19]. Borgatti and Everett's function takes as input a network and a putative division into core and periphery and returns a score that indicates whether the division is a "good" one in a certain sense. Then by maximizing the function over all possible divisions one can find the "best" core-periphery decomposition. Variants of the same idea have been explored by Rombach *et al.* [13], as well as by Kojaku and Masuda [20] who proposed a multi-group version. Cucuringu *et al.* [21] have explored several methods for finding core-periphery structure based on counting geodesic paths and on spectral network properties. Other methods use node-level properties of the network such as the clustering coefficient [12] or centrality measures [22].

Stochastic block models, commonly used in community detection [23, 24], have also been applied to core-periphery structure. In these models, one assumes that nodes are divided into types (e.g., core and periphery) and that the probability ω_{rs} of an edge between a pair of nodes depends on the types r and s of the nodes. Zhang *et al.* [25] studied a two-group version of this model with $\omega_{11} > \omega_{12} > \omega_{22}$, which generates networks with classic core-periphery structure, then fitted the model to empirical network data to detect structure. Gallagher *et al.* [26] took a similar approach but with more than two groups and multiple nested cores such that $\omega_{rs} = f[\max(r, s)]$ for some function $f[r]$.

In this paper we also take a model-fitting approach to the study of core-periphery structure, but we formulate a more general model that includes the classic two-group structure but also allows for more flexible structures as well. In particular, we consider a hierarchical model with any number of groups, the number being determined by the network structure using Bayesian model selection,

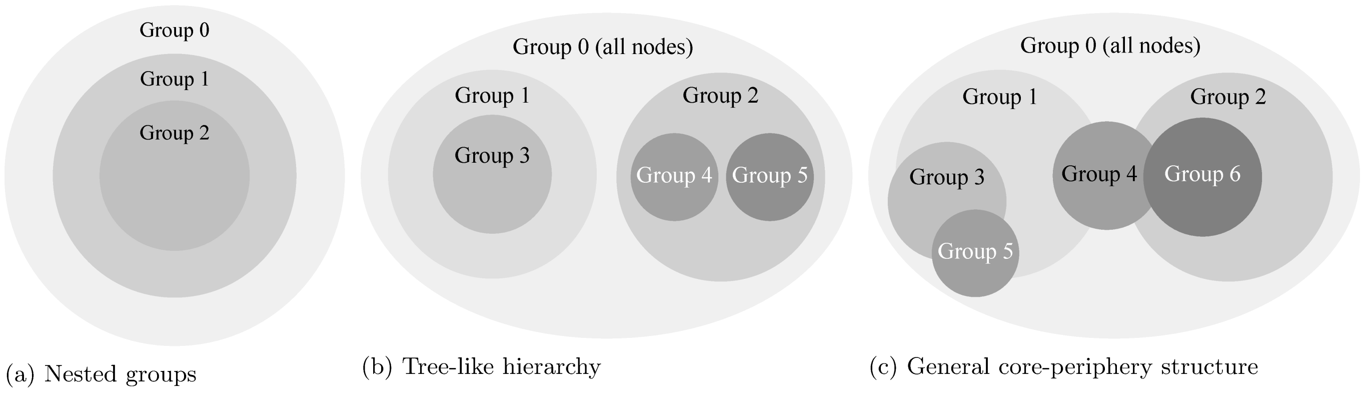


FIG. 1: The model proposed here is capable of capturing a range of different types of structure, including (a) traditional nested groups, (b) tree-like branching structures, and (c) complex structures of overlapping groups.

and we specifically allow for the possibility that the hierarchy may not be perfectly nested and moreover that there maybe multiple cores and multiple peripheries at any level in the hierarchy. These features allow our model to capture variants of core-periphery structure that occur in real-world networks but are not captured by traditional analyses. In a multilevel hierarchy, for instance, an inner core may not be perfectly contained within an outer one. In a network with community structure the individual communities may each have a separate core of their own, or there may be several cores within a single periphery. In this sense the structures we consider are not directly comparable to previously studied forms of core-periphery structure and we do not expect the output of our algorithms to be directly comparable. But by fitting our proposed model to network data we provide a way to search for unexpected details of network structure, and in the process encounter some new formations that have not been previously observed.

II. A HIERARCHICAL MODEL OF NETWORK STRUCTURE

One can think of conventional core-periphery structure as a hierarchy. In the simplest case the hierarchy has just two levels, an outer periphery and an inner core. In more elaborate cases there can be an onion-like succession of levels, each one nested inside the next, from the outermost to the innermost—see Fig. 1a. This, however, is not the most general kind of nested structure. A more general structure can have any number of first-level cores within a single periphery, and any number of second-level cores within those first-level ones, and so forth (Fig. 1b). A structure like this can be captured with a tree or dendrogram, and some attempts have been made to fit dendrograms to networks [5]. The work presented here goes still further by allowing for the possibility that the hierarchy is not perfectly nested. In our model, we envisage a hierarchical series of cores, but we allow these cores to be placed in any way around the network, with no requirement that they be nested within one another (Fig. 1c). In practice, if two cores do not overlap then they do not interact and hence their relative order in the hierarchy has no effect, but if they overlap then the higher ranked one takes precedence over the lower in a manner we will describe. The net result is a hierarchical model that generalizes both the conventional onion layers and the dendrogram and, as we will see, captures a wide range of possible structures. We then fit this model to network data to infer core-periphery structure in real-world networks. In detail the procedure is as follows.

Consider an unweighted, undirected network of n nodes, which can belong to any of k groups labeled by $r = 0 \dots k-1$. By contrast with traditional community structure models we allow nodes to belong to any number of groups simultaneously, up to and including all of them. In addition, every node always belongs to group 0, which acts as a sort of default or base group. Thus each node belongs to group 0 plus some selection of the other groups $1 \dots k-1$. (We label the first group 0 rather than 1 to remind ourselves of its special status.)

We define a set of indicator variables g_u^r such that $g_u^r = 1$ if node u belongs to group r and 0 otherwise. Then we define a set of probabilities ω_r, one for each group including group 0, and place edges between node pairs independently with probability ω_r, where r is the *highest common group* that both nodes of the pair belong to, meaning the one with the highest number. For example, if one node belongs to groups 0, 1, 2 and another belongs to 0, 1, 3, then there is an edge between them with probability ω_1.

Figure 1c illustrates the behavior of the resulting network: the groups form "patches" that lie one on top of another and the topmost visible patch takes precedence for each node pair. For instance, if all nodes belong to group 0 only, then every pair of nodes has equal probability ω_0 of being connected, which gives us a standard random graph. But if we assign some subset of the nodes to group 1 then for any pair of nodes that are both in group 1 the probability of an edge becomes ω_1 and overrides the previous ω_0. Similarly any pair of nodes assigned to group 2, including nodes already in groups 0 and 1, have probability ω_2 of connection, overriding ω_0 and ω_1, and so forth. The end result is a model that starts out as a random graph but then adds variation and detail to the network wherever it is needed to capture local structural features. This approach has some similarities to those of Kojaku and Masuda [20] and Gallagher *et al.* [26], but it differs crucially in that it does not force the groups to be either strictly nested within each other or non-overlapping. Given the definition of the model, our goal is now to fit it to observed network data to infer the best choice of groups g_u^r for each node.

Suppose we have a network of n nodes, represented by its $n \times n$ adjacency matrix A with elements $a_{uv} = 1$ if there is an edge between nodes u and v and 0 otherwise. Then the probability of observing a particular network if it was generated from our model with given k and given group memberships g is

$$\begin{aligned} P(A|\omega, k, g) &= \prod_{u<v} \omega_{h(u,v)}^{a_{uv}} \big[1 - \omega_{h(u,v)}\big]^{1-a_{uv}} \\ &= \prod_{r=0}^{k-1} \omega_r^{m_r} (1-\omega_r)^{t_r - m_r}, \end{aligned} \tag{1}$$

where $h(u,v)$ is the highest common group of nodes u and v, and

$$t_r = \sum_{u<v} \delta_{r,h(u,v)} \tag{2}$$

is the number of node pairs with highest common group r, and

$$m_r = \sum_{u<v} a_{uv} \delta_{r,h(u,v)} \tag{3}$$

is the number of such pairs that are connected by an edge.

Our primary interest in performing the fit is to determine the group memberships g_u^r. The values of the parameters ω_r are not of particular interest, so we eliminate them by marginalizing. We assume a uniform prior $P(\omega_r) = 1$ for all r and write

$$\begin{aligned} P(A|k, g) &= \int P(A, \omega|k, g)\, d\omega \\ &= \int P(A|\omega, k, g) P(\omega)\, d\omega \\ &= \prod_{r=0}^{k-1} \int_0^1 \omega_r^{m_r} (1-\omega_r)^{t_r - m_r}\, d\omega_r \\ &= \prod_{r=0}^{k-1} \frac{m_r!(t_r - m_r)!}{(t_r+1)!}. \end{aligned} \tag{4}$$

A. Prior on group assignments

We consider two different scenarios. In the first, the number k of groups in the network is fixed; in the second it is not. The number might be fixed if, for instance, our goal is to find traditional core-periphery structure, for which there are always two groups, the core and the periphery. In other cases, we may be interested in allowing the number of groups to vary and determining what number best fits the network we observe.

If the number of groups is fixed, we can write

$$P(g|A, k) = \frac{P(A|g, k) P(g|k)}{P(A|k)}, \tag{5}$$

and we can maximize this quantity to find the most probable values of g, or sample from the distribution it defines to generate plausible core-periphery structures in proportion to their probability. In this paper we do the latter, using a Monte Carlo method.

To do this we need to make a choice for the prior $P(g|k)$. Naively we might assume a prior that is uniform over all assignments g, but this would be a mistake. Such a choice produces group sizes that are narrowly peaked about $\frac{1}{2}n$, which is highly unrealistic. Similar problems occur in traditional community detection [27, 28], where a good solution is instead to choose the *sizes* of the groups to be uniform rather than the assignments and we take an analogous approach here. For each group $r > 0$ we first choose a size n_r uniformly at random between 0 and n, each value thus having probability $1/(n+1)$. Then we choose uniformly at random one of the $\binom{n}{n_r}$ ways to assign n_r nodes to the group, each choice thus having probability $1/\binom{n}{n_r}$. Hence for all groups the total probability of an assignment is

$$P(g|k) = \prod_{r=1}^{k-1} \frac{1}{(n+1)\binom{n}{n_r}} = \prod_{r=1}^{k-1} \frac{n_r!(n-n_r)!}{(n+1)!}. \tag{6}$$

B. Prior on number of groups

In traditional core-periphery structure calculations one considers the existence of a single core and a single periphery, and this is the approach taken for instance in [25]. If our goal, however, is to find multiple groups of unknown number, including multiple or overlapping cores, then we need to allow k to vary, which means choosing a prior on k. Here we adopt the approach taken in [29, 30] and use a Poisson distribution with mean 1:

$$P(k) = \frac{\mathrm{e}^{-1}}{(k-1)!}. \tag{7}$$

Note that group 0 always exists, so the distribution of the number of groups is effectively a distribution over $k-1$, which is why we have $1/(k-1)!$ in the denominator.

With this choice, we can now write

$$P(g,k|A) = P(k)P(g|A,k) = \frac{P(k)P(g|k)P(A|g,k)}{P(A)} \propto \frac{1}{(k-1)!}\prod_{r=1}^{k-1}\frac{n_r!(n-n_r)!}{(n+1)!}\prod_{r=0}^{k-1}\frac{m_r!(t_r-m_r)!}{(t_r+1)!}. \tag{8}$$

Again, we can sample from this probability to generate a selection of values k and group assignments g that are representative of the network. In the following section we describe the algorithm we use to achieve this.

III. SAMPLING FROM THE POSTERIOR DISTRIBUTION

Our goal is to sample from the posterior distribution (8), which we do using a Markov-chain Monte Carlo method. This procedure effectively generates a series of plausible core-periphery decompositions of the given network, not a single decomposition, although in practice one can simply examine the highest-probability state found as an indication of the structure present in the network. We describe the method first for the simpler case where the number of groups k is fixed, then for the more complicated case of varying k.

A. Monte Carlo algorithm for the case of fixed k

For the case of fixed k our Monte Carlo scheme is as follows.

1. We choose a group s uniformly at random from $1\ldots k-1$.

2. With equal probability $\frac{1}{2}$ we propose to either remove a node from group s or add a node to it. If we are removing, the node to be removed is chosen uniformly at random from those currently in the group. If there are no nodes in the group, we do nothing and move on to the next Monte Carlo step. If we are adding, the node to be added is chosen uniformly at random from those currently not in the group. If the group is full—all n nodes are already members—we do nothing and move on to the next step.

3. The proposed move is accepted with the Metropolis-Hastings style acceptance probability

$$\alpha(g\to g') = \min\biggl(1, \frac{P(A|g',k)}{P(A|g,k)}\biggr), \tag{9}$$

where g' represents the group assignments after the addition or removal. If the move is accepted, the chosen node is added or removed as proposed. If the move is not accepted, the group assignments g remain unchanged on this step.

4. Repeat from step 1.

In the limit where this algorithm tends to an equilibrium distribution of states, that distribution will be the one given in Eq. (5). To demonstrate this, it suffices to prove two results: first that the algorithm is ergodic and second that it satisfies detailed balance. Ergodicity requires that every state of the system be accessible from every other by a finite sequence of moves. This is trivially true in the present case, since the membership of any group can be set to anything we like in at most n moves by first removing any nodes we don't want and then adding in those we do.

Detailed balance is a little more complicated. Detailed balance requires that in equilibrium the average rate of moves $g\to g'$ equals the average rate $g'\to g$, which means

$$P(g|A,k)P(g\to g') = P(g'|A,k)P(g'\to g), \tag{10}$$

where $P(g\to g')$ is the probability of making the transition $g\to g'$. This probability can be written as

$$P(g\to g') = \pi(g\to g')\alpha(g\to g'), \tag{11}$$

where $\pi(g\to g')$ is the probability of proposing the move and $\alpha(g\to g')$ is the probability of accepting it as in Eq. (9). Then Eq. (10) can be written as

$$\frac{P(g'|A,k)}{P(g|A,k)} = \frac{\pi(g\to g')\,\alpha(g\to g')}{\pi(g'\to g)\,\alpha(g'\to g)}. \tag{12}$$

We can show that this condition is satisfied by the proposed Monte Carlo algorithm as follows.

From Eq. (5) we have

$$\frac{P(g'|A,k)}{P(g|A,k)} = \frac{P(A|g',k)P(g'|k)}{P(A|g,k)P(g|k)}, \tag{13}$$

while from Eq. (9) the ratio of the two acceptance probabilities is

$$\frac{\alpha(g\to g')}{\alpha(g'\to g)} = \frac{P(A|g',k)}{P(A|g,k)}. \tag{14}$$

Substituting (13) and (14) into (12), a factor of $P(A|g',k)/P(A|g,k)$ cancels and we are left with

$$\frac{P(g'|k)}{P(g|k)} = \frac{\pi(g\to g')}{\pi(g'\to g)}. \tag{15}$$

If our Monte Carlo algorithm satisfies this condition, then it satisfies detailed balance.

Using Eq. (6), the left-hand side can be written as

$$\frac{P(g'|k)}{P(g|k)} = \prod_{r=1}^{k-1}\frac{n'_r!(n-n'_r)!}{n_r!(n-n_r)!}, \tag{16}$$

where n_r is the number of nodes in group r before the move and n'_r is the number afterwards. Suppose the particular move we are considering $g\to g'$ is one that adds a node to group s. Then $n'_s = n_s + 1$, while $n'_r = n_r$ for all other groups, so (16) simplifies to

$$\frac{P(g'|k)}{P(g|k)} = \frac{(n_s+1)!(n-n_s-1)!}{n_s!(n-n_s)!} = \frac{n_s+1}{n-n_s}. \tag{17}$$

For the right-hand side of Eq. (15), for the same move that adds a node to group s, the proposal probability is

$$\pi(g \to g') = \frac{1}{k-1} \times \frac{1}{2} \times \frac{1}{n-n_s} = \frac{1}{2(k-1)(n-n_s)}. \tag{18}$$

Here the factor $1/(k-1)$ is the probability of choosing the particular group r out of all $k-1$ possibilities, the factor $\frac{1}{2}$ is the probability of choosing to add a node, and the factor $1/(n-n_s)$ is the probability of choosing the particular node to be added from the $n-n_s$ possibilities. Meanwhile, for the reverse move $g' \to g$, which involves removing the same node from group s again, the proposal probability is

$$\pi(g' \to g) = \frac{1}{k-1} \times \frac{1}{2} \times \frac{1}{n_s+1} = \frac{1}{2(k-1)(n_s+1)}, \tag{19}$$

since there are now n_s+1 nodes in the group. The ratio of the two proposal probabilities is thus

$$\frac{\pi(g \to g')}{\pi(g' \to g)} = \frac{1/2(k-1)(n-n_s)}{1/2(k-1)(n_s+1)} = \frac{n_s+1}{n-n_s}, \tag{20}$$

which agrees with Eq. (17) and hence Eq. (15) is satisfied and detailed balance is obeyed in this instance. The proof for the case where we remove a node from group s follows the same lines and leads to the same conclusion: the algorithm satisfies detailed balance and hence samples correctly from the target distribution $P(g|A,k)$ in equilibrium.

B. Algorithm for varying k

When k is allowed to vary the algorithm is more complex, involving two types of moves that each take us from a combined state (g,k) to a state (g',k'), as follows.

Type 1: In a move of type 1 we choose a group s uniformly at random from $1\dots k-1$. With probability $\frac{1}{2}$ we add a new node to the group chosen uniformly from the set of nodes that do not currently belong; otherwise, we remove an existing node from the group, chosen uniformly from those in the group. If we choose to add a node but group s is already full then we do nothing and move on to the next Monte Carlo step. If we choose to remove a node and group s is already empty then the entire group is deleted and the number of groups k decreases by one, with the labels of all groups above s also decreasing by one so that they still run to a maximum of $k-1$.

Type 2: In a move of type 2 we choose a group index s uniformly at random from $1\dots k$. We increase by one the labels of all groups s and greater (if there are any), create a new empty group with label s, and increase the value of k by one.

With these definitions, the complete algorithm is now as follows:

1. With probability $1 - 1/2k(n+1)$ propose a move of type 1.
   - 1a) If $k=1$ do nothing, since this implies all nodes are in group 0 only, so there are no moves to be made and there is no change of state on this Monte Carlo step.
   - 1b) Otherwise when $k>1$ choose a random move of type 1.
2. Else, with probability $1/2k(n+1)$, choose a random move of type 2.
3. Accept the proposed move with probability
$$\alpha(g,k \to g,k) = \min\biggl(1, \frac{P(A|g',k')}{P(A|g,k)}\biggr). \tag{21}$$
   Accepted moves are performed as proposed. If the move is not accepted the state of the system remains unchanged.
4. Repeat from step 1.

This algorithm again satisfies the condition of ergodicity trivially: we can reach any state with any number of groups in a finite number of moves by first removing all nodes from all groups except group 0, then removing the groups themselves, then adding back the appropriate number of groups and filling them with the desired nodes. The algorithm also satisfies the condition of detailed balance, which for this algorithm takes the form

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

The left-hand side can be written as

$$\frac{P(g',k'|A)}{P(g,k|A)} = \frac{P(g',k')P(A|g',k')}{P(g,k)P(A|g,k)}, \tag{23}$$

and the ratio of acceptance probabilities is

$$\frac{\alpha(g,k \to g',k')}{\alpha(g',k' \to g,k)} = \frac{P(A|g',k')}{P(A|g,k)}. \tag{24}$$

Substituting from (23) and (24) into (22), a factor of $P(A|g',k')/P(A|g,k)$ cancels and our detailed balance condition reduces to

$$\frac{P(g',k')}{P(g,k)} = \frac{\pi(g,k \to g',k')}{\pi(g',k' \to g,k)}. \tag{25}$$

From Eqs. (6) and (7) the left-hand side is

$$\frac{P(g',k')}{P(g,k)} = \frac{(k-1)!\prod_{r=1}^{k-1} n'_r!(n-n'_r)!/(n+1)!}{(k'-1)!\prod_{r=1}^{k'-1} n_r!(n-n_r)!/(n+1)!}. \tag{26}$$

Consider first the case where we propose a move of type 1 that adds a node to group s. Then $k' = k$ and $n'_s = n_s+1$, and $n'_r = n_r$ for all other groups r, so Eq. (26) becomes

$$\frac{P(g',k')}{P(g,k)} = \frac{n_s+1}{n-n_s}. \tag{27}$$

The probability of proposing such a move is

$$\begin{aligned}\pi(g,k\to g',k') \\ &= \biggl(1-\frac{1}{2k(n+1)}\biggr)\times\frac{1}{k-1}\times\frac{1}{2}\times\frac{1}{n-n_s} \\ &= \biggl(1-\frac{1}{2k(n+1)}\biggr)\frac{1}{2(k-1)(n-n_s)}, \end{aligned}\tag{28}$$

while the probability of proposing the reverse move is

$$\pi(g',k'\to g,k) = \biggl(1-\frac{1}{2k(n+1)}\biggr)\frac{1}{2(k-1)(n_s+1)}. \tag{29}$$

Thus the ratio of the two is

$$\frac{\pi(g,k\to g',k')}{\pi(g',k'\to g,k)} = \frac{n_s+1}{n-n_s}. \tag{30}$$

Between Eqs. (27) and (30), our detailed balance condition (25) is now satisfied. By a similar argument we can show that detailed balance is also satisfied when a node is removed from a group.

Now consider a move of type 2, which creates a new empty group with a random label s. For such a move Eq. (26) becomes

$$\begin{aligned}\frac{P(g',k')}{P(g,k)} &= \frac{(k-1)!\prod_{r=1}^{k} n'_r!(n-n'_r)!/(n+1)!}{k!\prod_{r=1}^{k-1} n_r!(n-n_r)!/(n+1)!} \\ &= \frac{1}{k(n+1)}\,, \end{aligned}\tag{31}$$

where all factors inside the products have canceled except for those pertaining to the new group, which gives us the factor of $1/(n+1)$.

The proposal probability for this move is equal to the probability that we decide to do a move of type 2 times the probability that we choose to add a new group with a particular label s out of the k possibilities, giving

$$\pi(g,k\to g',k') = \frac{1}{2k(n+1)}\times\frac{1}{k} = \frac{1}{2k^2(n+1)}. \tag{32}$$

The reverse move on the other hand occurs when we perform a move of type 1 and choose group s from the k possibilities, then attempt to remove a node only to discover that the group is already empty, causing us to delete the entire group. The proposal probability for this move is

$$\begin{aligned}\pi(g',k'\to g,k) &= \biggl(1-\frac{1}{2(k+1)(n+1)}\biggr)\times\frac{1}{k}\times\frac{1}{2} \\ &= \frac{1}{2k}\biggl(1-\frac{1}{2(k+1)(n+1)}\biggr). \end{aligned}\tag{33}$$

Now the ratio of the two probabilities is

$$\begin{aligned}\frac{\pi(g,k\to g',k')}{\pi(g',k'\to g,k)} &= \frac{4k(k+1)(n+1)/(2(k+1)(n+1)-1)}{2k^2(n+1)} \\ &= \frac{2(k+1)/k}{2(k+1)(n+1)-1} \\ &= \frac{1}{k(n+1)} + \mathrm{O}(1/n^2). \end{aligned}\tag{34}$$

Here we assume n is large and hence that terms of order $1/n^2$ can be neglected, making Eq. (34) equal to Eq. (31), and hence our detailed balance condition, Eq. (25), is satisfied.

This completes the proof of correctness of our algorithms. In the following sections we apply these algorithms to fit our model to a variety of networks in order to study core-periphery structure.

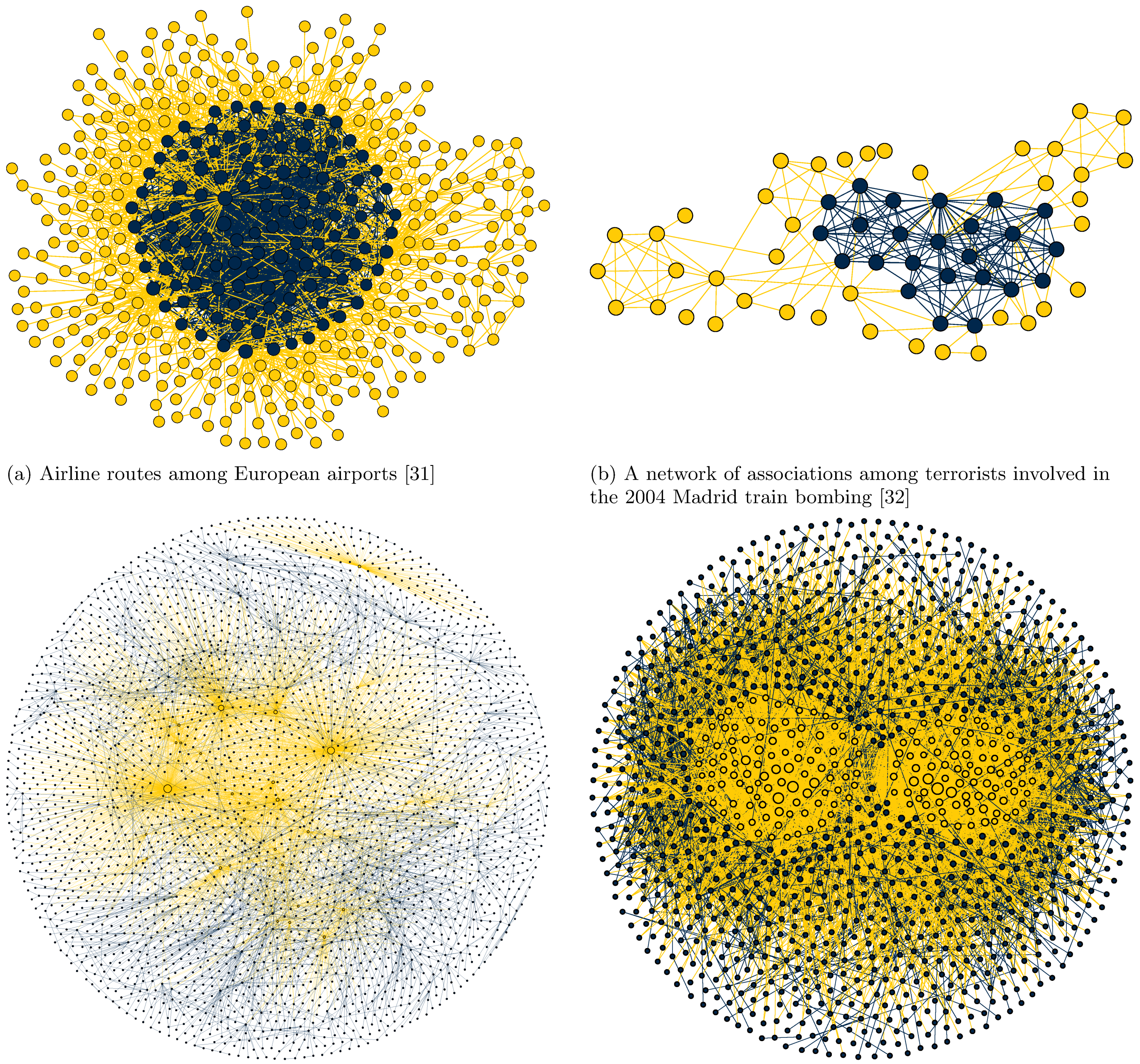
C. Implementation

The algorithms described above are reasonably efficient. The bottlenecks for run time are the evaluation of the quantities t_r, Eq. (2), and—for the algorithm with varying numbers of groups—the addition and removal of groups.

When a node u is added to or removed from group r, the highest common group between u and any of the other members of r may change, requiring updates to the variables t_r. In the present implementation we iterate individually over all nodes in r to check for changes. The highest common group between two nodes can be calculated in constant time, so this procedure takes time of order the average size of a group, which is n/k. For the version of the algorithm in which k is fixed, this gives a contribution of order n to the run time per Monte Carlo step.

For the version of the algorithm with varying number of groups, there is extra work to be done when the number of groups changes. As described in Section III B, when we delete a group, the labels of all nodes in higher-numbered groups must be decreased by one, which takes worst-case time $\mathrm{O}(n)$. On the other hand, moves that decrease the number of groups are relatively rare, occurring approximately k/n of the time, so that on average the relabeling of nodes contributes time $\mathrm{O}(n)\times k/n = \mathrm{O}(k)$ to the run time.

Overall, therefore, we expect the run time per step to be $\mathrm{O}(n+k)$, or $\mathrm{O}(n)$ if we assume that n increases faster than k, and numerical tests confirm this behavior. Assuming that the number of Monte Carlo steps for the entire calculation scales as the number of nodes, this implies the total run time will be $\mathrm{O}(n^2)$, which is workable though not ideal: many community detection algorithms run in $\mathrm{O}(n^2)$ time, but the fastest, such as the Louvain algorithm [35], run in time $\mathrm{O}(n\log n)$. It is possible that one might be able to find a shortcut that would allow one to compute updates to the t_r faster than $\mathrm{O}(n)$, but we



(a) Airline routes among European airports [31]

(b) A network of associations among terrorists involved in the 2004 Madrid train bombing [32]

(c) Network representation of the Internet in November 1997 at the autonomous system level [33]

(d) A network of hyperlinks among a set of US political blogs [34]

FIG. 2: Two types of two-group core-periphery structure distinguished by the hierarchical model. Panels (a) and (b) have a core that is densely connected within itself but only sparsely connected to the periphery. This is the traditional definition of core-periphery structure. Panels (c) and (d) on the other hand show a kind of "inside-out" structure in which the core is strongly connected within itself and strongly connected to the periphery.

do not at present know of a way to do this.

Our current implementation of the algorithm performs about 500 000 Monte Carlo steps per second for the smaller networks on standard hardware (circa 2023), dropping to 100 000 or less for the largest ones, with total running times ranging from a few minutes to about two hours for the example networks studied here. We do not give any direct comparisons of run time or performance of our algorithm against other algorithms for core-periphery structure in this paper, since the particular type of structure we consider is different from other types that go under the "core-periphery" name. As far as we are aware, there are no competing methods for finding hierarchical structure of the kind described in this paper.

IV. EXAMPLE APPLICATIONS

In this section we give example applications of our methods to a selection of real-world networks, revealing a range of behaviors and structures of interest in the core-periphery divisions of these systems.

A. Traditional two-group core-periphery structure

For our first set of examples, we perform calculations in which the number of groups is fixed at $k = 2$, which corresponds to the traditional two-group core-periphery structure, as studied by many previous authors. Figure 2 shows examples of such structure found in four different networks. For each network the structure shown is the highest-probability structure found during a single run of our algorithm with 10^9 Monte Carlo steps and in each case the algorithm finds clear core-periphery divisions, as highlighted by the colors.

A number of interesting features emerge in these examples. First, we note that, as our model is defined, it is arguably the *edges* that belong to groups, not the nodes. As we have said, a node can belong to any number of groups, but an edge only belongs to one: the properties of each edge are determined solely by the highest-numbered group to which its two nodes both belong and in this sense the edge belongs to this group only. In Fig. 2 we have colored the edges according to the group they belong to and this provides a clear and useful visualization. (The same trick will also be useful in Section IV B when we study divisions with larger numbers of groups.)

All the images in Fig. 2 use the same color scheme: group 0 is in yellow and group 1 is in blue. The figure reveals that there are two distinctly different types of core-periphery structure, one where the core is group 0 and one where it is group 1. Recall that the probability ω_r of connection between two nodes depends on their highest common group r, meaning in this case that edges between nodes that are both in group 1 have probability ω_1 while all others have probability ω_0. With this in mind take a look at the figure.

Figures 2(a) and (b) show results for a network of airline routes [31] and a network of associations among a group of terrorists [32] respectively. In both of these cases the core found in the network is represented by group 1 (in blue) and the periphery by group 0 (in yellow), with $\omega_1 > \omega_0$. This implies that there is a high probability of edges within the core (blue edges) and a lower probability both in the periphery and also between the core and the periphery (yellow edges).

Conversely, in Figs. 2(c) and (d), which represent the Internet at the autonomous system level [33] and a network of political weblogs [34], the groups are reversed, with the core being group 0 and the periphery being group 1, and $\omega_1 < \omega_0$. In this "inside-out" type of structure there is a high probability of connections both within the core and between the core and periphery (yellow edges), and a lower probability in the periphery (blue edges).

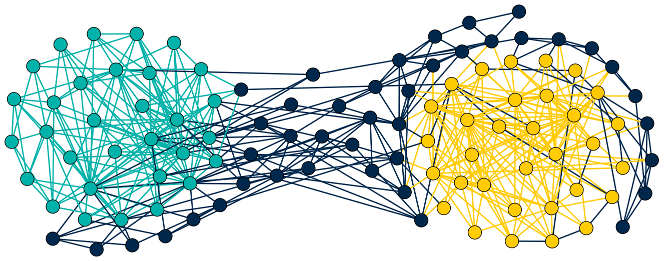


FIG. 3: Political books network with a periphery and two cores corresponding to left and right leaning books.

These two types of core-periphery structure represent quite different circumstances. In the first, the core is isolated from the periphery in the sense that it is densely connected only within itself and sparsely connected to everything else. In the second, the core is strongly connected everywhere, both to itself and to others and dominates the connectivity of the network. The latter ("inside-out") structure is particularly interesting because it deviates from the traditional definition of core-periphery structure as formulated for instance by Borgatti and Everett [11], who assumed an isolated core. Our method naturally and automatically distinguishes between the two types of structure.

The two types make some sense in the present case. For the airline route network, for instance, the core broadly represents airline hubs and the periphery represents regional airports. One expects strong connections between hubs—almost all pairs of hubs have direct flights—but one expects only weak connections to the outlying airports, many of which only fly to a single hub. Conversely, in the weblog network, for example, the core represents the most influential blogs, ones which most members of the community link to, so we expect connections to be strong not only within the core but also between the core and the periphery.

B. Structure with an arbitrary number of groups

Now let us look at what happens when we allow the number of groups to vary, taking whatever value is necessary to best fit the structure of the network. Here again we find some interesting features. As a first example, Fig. 3 shows a copurchasing network of books. The nodes in this network represent 105 popular books on US politics and the edges represent frequent copurchase on Amazon.com, i.e., purchase by the same buyers. This network, which has been studied previously by a number of authors [36, 37], is known to show clear community structure in which the network divides into communities of left- and right-leaning books. Our core-periphery analysis, as indicated by the colors in the figure, finds three groups: two cores and a single periphery. The two

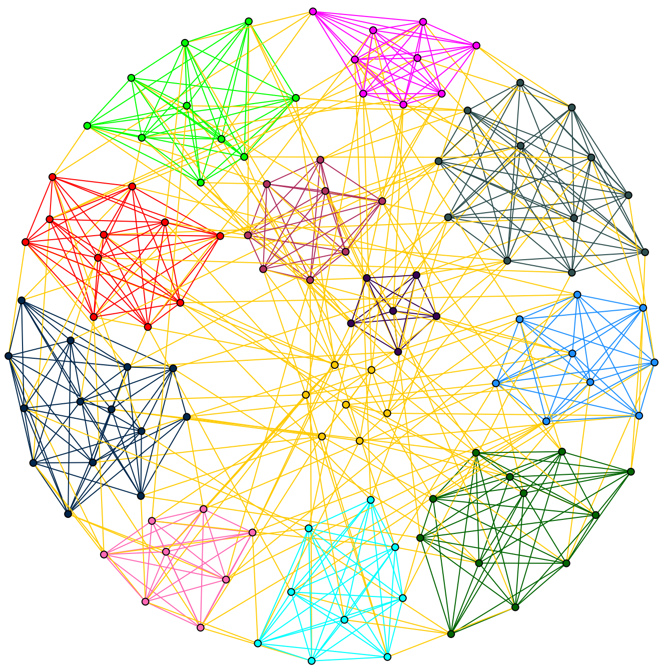
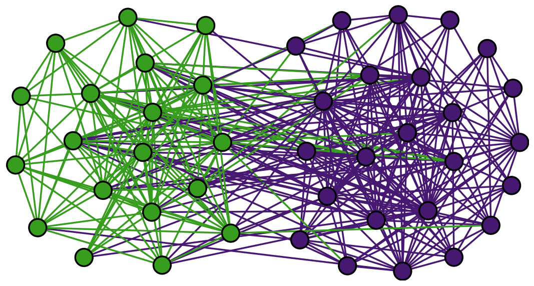
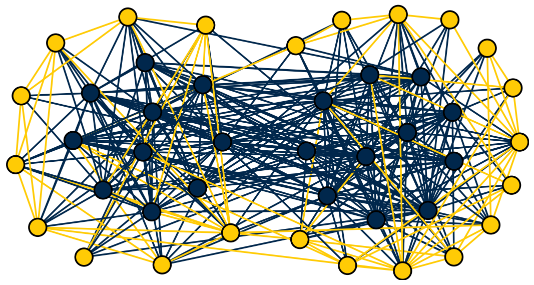


FIG. 4: American Football network where each of the cliques are connected to each other via the periphery.



(a) Traditional community structure



(b) Core-periphery structure

FIG. 5: Structure found in the network of windsurfers.

cores correspond to the innermost members of the left- and right-leaning communities while the periphery captures the remainder of the network. Thus, the algorithm has found the political divide between left and right but also finds a large group of peripheral books that, at least in this analysis, are well represented as a homogeneous mass, suggesting that they are not strongly connected to either side of the political aisle.

Figure 4 shows a similar finding for another well studied example of community structure, a network of competition between US college teams in the sport of American football [3]. College football teams are divided into a number of groups or “conferences,” and most games are played between teams in the same conference, so the network of games played, as analyzed here, has strong community structure which can easily be discovered with a range of community detection algorithms. Again, however, our core-periphery analysis returns a more subtle picture, as shown in the figure. Our algorithm finds a separate core for each conference, accurately dividing most teams into the 11 conferences in the network. A small number of teams—many of them independents who belong to no conference—are not assigned to any core, and all inter-conference games are assigned to the periphery. This makes good sense: it tells us that the conferences constitute a clear set of separate groups in the network, while inter-conference play and non-conference teams constitute a single periphery. This is an accurate description of the network and a more economical one than the standard community structure division, as found for instance using the stochastic block model, which also assigns a separate community for each conference but in addition assigns a separate probability for inter-conference play between every single pair of conferences, rather than recognizing that a single periphery is an adequate and more parsimonious description.

In these last two examples our algorithm has found a hybrid of core-periphery structure and community structure. While this is illuminating for these particular examples, it is important to realize that this is not inevitable, and that the algorithm will return other structures where appropriate. Figure 5 shows an example. The network in this figure is a famous one from the social networks literature, a network of interactions observed by Freeman [38] between a group of people windsurfing off the California coast in 1986. This network is known to have a clear two-group community structure which is easily found by community detection—see Fig. 5a. When analyzed using the methods of this paper we also find two groups, but they are not the same: now we find core and periphery but no clear division between the communities, suggesting that connections within the core may be just as important as divisions between the two communities.

V. CONCLUSIONS

In this paper we have proposed a hierarchical model of core-periphery structure in networks and a Monte Carlo scheme for fitting it to observed network data. Applying these methods to a variety of real-world networks we find a number of interesting patterns. The method is able to capture traditional two-group core-periphery structure consisting of a dense core weakly connected to a sparse periphery. In some networks, however, we find that a better fit is given by a novel "inside-out" structure in which the core is connected strongly both within itself and to the periphery. Various networks are better represented by one or other of the two types of structure and the distinction between the two could offer a more nuanced view of structure and function in these networks.

We have also investigated cases where there are more than two groups in the network, generalizing the traditional core-periphery structure (as other authors have also done). For this we use a Monte Carlo scheme that allows the number of groups to vary freely, automatically choosing the number that best fits the network in question. In some cases, we find a structure akin to a hybrid between core-periphery structure and community structure in which there is a separate core in each of several communities plus a single periphery surrounding all of them. In other cases, we find pure core-periphery structure without any communities.

There are a number of possible directions for further research using these methods. First, we have looked here at only the highest probability structures found by our algorithms but in principle the algorithms return a complete sample of high-probability structures drawn from the posterior distribution of the model and it would be interesting to study the range of structures within such a sample. Are they all closely similar, so that a single consensus structure can well represent them all, or is there significant variation between structures, and if so of what kind? Second, one could examine generalizations of the method to broader classes of networks, such as directed and weighted networks and multiplex networks. Another interesting question is whether there exists a natural "degree-corrected" version of the model akin to the degree-corrected stochastic block model of [24]. The model proposed here is not degree corrected, which could cause issues with networks that have a very broad degree distribution. These questions, however, we leave for future work.

Acknowledgments

The authors thank the Simons Institute for the Theory of Computing at the University of California, Berkeley for hospitality while part of this work was conducted. This work was funded in part by the US National Science Foundation under grant DMS–2005899. Computer code implementing the methods of this paper is available at `https://github.com/apolanco115/hcp`.

[1] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, Complex networks: Structure and dynamics. *Physics Reports* **424**, 175–308 (2006).

[2] M. Newman, *Networks.* Oxford University Press, Oxford, 2nd edition (2018).

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

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

[5] A. Clauset, C. Moore, and M. E. J. Newman, Hierarchical structure and the prediction of missing links in networks. *Nature* **453**, 98–101 (2008).

[6] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435**, 814–818 (2005).

[7] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, Mixed membership stochastic blockmodels. *Journal of Machine Learning Research* **9**, 1981–2014 (2008).

[8] P. D. Hoff, A. E. Raftery, and M. S. Handcock, Latent space approaches to social network analysis. *J. Amer. Stat. Assoc.* **97**, 1090–1098 (2002).

[9] M. E. J. Newman and T. P. Peixoto, Generalized communities in networks. *Phys. Rev. Lett.* **115**, 088701 (2015).

[10] W. L. Hamilton, R. Ying, and J. Leskovec, Representation learning on graphs: Methods and applications. In *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, Institute of Electrical and Electronics Engineers, New York (2017).

[11] S. P. Borgatti and M. G. Everett, Models of core/periphery structures. *Social Networks* **21**, 375–395 (1999).

[12] P. Holme, Core-periphery organization of complex networks. *Phys. Rev. E* **72**, 046111 (2005).

[13] M. P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha, Core-periphery structure in networks. *SIAM J. Appl. Math.* **74**, 167–190 (2014).

[14] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir, A model of Internet topology using k-shell decomposition. *Proc. Natl. Acad. Sci. USA* **104**, 11150–11154 (2007).

[15] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani, K-core decomposition of internet graphs: Hierarchies, self-similarity and measurement biases. *Networks and Heterogeneous Media* **2**, 371–393 (2008).

[16] D. S. Bassett, N. F. Wymbs, M. P. Rombach, M. A. Porter, P. J. Mucha, and S. T. Grafton, Task-based core-periphery organization of human brain dynamics. *PLOS Computational Biology* **9**, 1–16 (2013).

[17] G. Fagiolo, J. Reyes, and S. Schiavo, The evolution of the world trade web: A weighted-network analysis. *Journal of Evolutionary Economics* **20**, 479–514 (2010).

[18] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani, K-core decomposition: A tool for the visualization of large scale networks. Preprint arxiv:cs/0504107 (2006).

[19] M. E. J. Newman and M. Girvan, Finding and evaluating community structure in networks. *Phys. Rev. E* **69**, 026113 (2004).

[20] S. Kojaku and N. Masuda, Finding multiple core-periphery pairs in networks. *Phys. Rev. E* **96**, 052313 (2017).

[21] M. Cucuringu, P. Rombach, S. H. Lee, and M. A. Porter, Detection of core-periphery structure in networks using

spectral methods and geodesic paths. *European Journal of Applied Mathematics* **27**, 846–887 (2016).

[22] S. H. Lee, M. Cucuringu, , and M. A. Porter, Density-based and transport-based core-periphery structures in networks. *Phys. Rev. E* **89**, 032810 (2014).

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

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

[25] X. Zhang, T. Martin, and M. E. J. Newman, Identification of core-periphery structure in networks. *Phys. Rev. E* **91**, 032803 (2015).

[26] R. J. Gallagher, J.-G. Young, and B. F. Welles, A clarified typology of core-periphery structure in networks. *Science Advances* **7**, eabc9800 (2021).

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

[28] M. A. Riolo, G. T. Cantwell, G. Reinert, and M. E. J. Newman, Efficient method for estimating the number of communities in a network. *Phys. Rev. E* **96**, 032310 (2017).

[29] 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).

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

[31] A. Cardillo, J. Gómez-Gardeñes, M. Zanin, M. Romance, D. Papo, F. d. Pozo, and S. Boccaletti, Emergence of network features from multiplexity. *Scientific Reports* **3**, 1344 (2013).

[32] B. Hayes, Connecting the dots: Can the tools of graph theory and social-network studies unravel the next big plot? *American Scientist* **94**(5), 400–404 (2006).

[33] J. Leskovec and A. Krevl, SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data` (2014).

[34] L. A. Adamic and N. Glance, The political blogosphere and the 2004 US election. In *Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem*, Association of Computing Machinery, New York (2005).

[35] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, Fast unfolding of communities in large networks. *J. Stat. Mech.* **2008**, P10008 (2008).

[36] M. E. J. Newman, Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA* **103**, 8577–8582 (2006).

[37] T. P. Peixoto, Revealing consensus and dissensus between network partitions. *Phys. Rev. X* **11**, 021003 (2021).

[38] L. C. Freeman, S. C. Freeman, and A. G. Michaelson, On human social intelligence. *Journal of Social and Biological Structures* **11**, 415–425 (1988).