University of Pennsylvania
UPenn Biostatistics Working Papers

Year 2007 *Paper* 22

Vertex Clustering in Random Graphs via Reversible Jump Markov Chain Monte Carlo

Stefano Monni* Hongzhe Li†

*University of Pennsylvania, smonni@mail.med.upenn.edu

†hongzhe@mail.med.upenn.edu

Vertex Clustering in Random Graphs via Reversible Jump Markov Chain Monte Carlo

Stefano Monni and Hongzhe Li

Abstract

Networks are a natural and effective tool to study relational data, in which observations are collected on pairs of units. The units are represented by nodes and their relations by edges. In biology, for example, proteins and their interactions, and, in social science, people and inter-personal relations may be the nodes and the edges of the network. In this paper we address the question of clustering vertices in networks, as a way to uncover homogeneity patterns in data that enjoy a network representation. We use a mixture model for random graphs and propose a reversible jump Markov chain Monte Carlo algorithm to infer its parameters. Applications of the algorithm to one simulated data set and three real data sets, which describe friendships among members of a University karate club, social interactions of dolphins, and gap junctions in the C. Elegans, are given.

Vertex Clustering in Random Graphs via Reversible Jump Markov Chain Monte Carlo

Stefano Monni*and Hongzhe Li

Department of Biostatistics and Epidemiology

University of Pennsylvania

1401C and 920 Blockley Hall, 423 Guardian Drive

Philadelphia, PA 19104-6021

USA

December 4, 2007

*smonni@mail.med.upenn.edu

Abstract

Networks are a natural and effective tool to study relational data, in which observations are collected on pairs of units. The units are represented by nodes and their relations by edges. In biology, for example, proteins and their interactions, and, in social science, people and inter-personal relations may be the nodes and the edges of the network. In this paper we address the question of clustering vertices in networks, as a way to uncover homogeneity patterns in data that enjoy a network representation. We use a mixture model for random graphs and propose a reversible jump Markov chain Monte Carlo algorithm to infer its parameters. Applications of the algorithm to one simulated data set and three real data sets, which describe friendships among members of a University karate club, social interactions of dolphins, and gap junctions in the C. Elegans, are given.

Key Words: Networks; Graphs; Modules; Clustering; Reversible Jump Markov Chain Monte Carlo; Wang Landau

1 Introduction

For many years graphs have been an important tool to study complex interacting systems appearing in different branches of sciences, from social sciences to biology, see *e.g.* the review (Albert and Barabási, 2002). A graph is a pair of two sets $(\mathcal{V}, \mathcal{E})$: elements of $\mathcal{V}$ are called vertices or nodes, and elements of $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are pairs of vertices, called edges. In the following, we consider only simple undirected graphs: namely, each element $e \in \mathcal{E}$ corresponds to one and only one unordered pair (i, j) of distinct points of $\mathcal{V}$. A matrix X, called the adjacency matrix, can be used to describe the graph. Its order is the cardinality of the set $\mathcal{V}$, the number of vertices, and its ij entry is non-zero ($X_{ij} = 1$) when the pair (i, j) is an edge. With the above assumptions, the adjacency matrix is binary-valued, symmetric, and its elements along the diagonal are zeros. When one represents the data set of a system with a graph, one associates

the elements of the system with vertices, and their relations with edges. The data set is however a sample from an unknown distribution, which is ultimately what one is interested in. Accordingly, the graph too should be considered just as a sample from some graph distribution, which would describe the data distribution. Stochastic graph models thus appear to be the natural framework to carry out data analysis where a graphical representation is needed. Eördos and Rényi (1959) introduced the concept of random graph; in what is now called the Eördos-Rényi (ER) model, each pair of vertices has the same probability of being an edge: $P(X_{ij} = 1) = p$ (Gilbert, 1959). The ER model has very often been used as a null model of randomness: the deviation from randomness of a real data network is established by comparing some of its topological properties with those of the ER network. This approach, *i.e.* using the ER network or any other random network with specified properties as a null model, is very common in the study of real data sets and has lead to the development of important concepts, such as that of scale-free network (Barabási and Albert, 1999). Measures used to evaluate the deviation from a null model include the degree distribution, the characteristic path length, and the clustering coefficient (Watts and Strogatz, 1998). One definition of the latter (Watts and Strogatz, 1998) is the ratio of the neighbors of a vertex to the maximum possible number of neighbors, averaged over all vertices in the network. The clustering coefficient underpins one concept of cluster as a group of vertices that are connected to each other more than they are connected to vertices outside the cluster. Many papers have studied the problem of identifying clusters, which in fact can provide deep insights into how the system represented by the network functions. The term module in biological applications or community in the social sciences is also employed, although there are various definitions of what a module or a community is, which differ in the underlying measure used. We refer the interested reader to (Danon *et al.*, 2005) for a comparison of some algorithms for community detection and some pointers to the literature.

This paper too deals with the important question of clustering vertices, in the general sense of uncovering some homogeneity in the heterogeneity of the population of

vertices. To this end, finite mixture distributions (Titterington *et al.*, 1985) appear to be the right framework. Recently, Daudin *et al.* (2006) proposed a finite mixture model for random graphs. To estimate the model parameters, they used an EM algorithm, which is approximate in its E-step: the joint distribution of the random variables that describe each the group to which a vertex belongs, is approximated by the product of the conditional distribution of each variable given the rest, and the latter are fixed to their conditional expectations. While the approximation is entirely justifiable because of the complexity of the distributions in the model, it raises some questions about the convergence of the algorithm or the dependence of its solutions on the starting points of the iterations. Moreover, they resorted to a heuristic criterion to choose the number of clusters. Another recent paper (Newman and Leicht, 2007) considers too a finite mixture model, and it employs an EM algorithm to determine the model parameters, under the assumption that the number of groups is known. A further study that deals with the problem of clustering in networks is by Handcock *et al.* (2007). They propose a model for networks, called latent position cluster model, where each node is assigned a latent position in a Euclidean space, which is drawn from a finite mixture of multi-variate normals. The probability for a pair of nodes to be an edge is assumed to be a function of the Euclidean distance of the nodes and is modeled by a logistic regression. Two methods are presented to determine the latent position and the cluster membership of the nodes: one is a two-stage maximum-likelihood estimation, whose second stage is implemented using the EM algorithm; the other is a standard Markov chain, which is shown to be the best performer of the two methods, as one should expect. The parameters are estimated for some models, *i.e.* for some choice of numbers of mixture components, and the best of them is selected via approximated Bayes factors.

Here, we consider a finite mixture model for random graphs and propose an algorithm based on the reversible jump Markov chain Monte Carlo (RJMCMC) method of Green (1995) and on the sampler of Wang and Landau (WL) (2001), as described in the context of RJMCMC by Atchadé and Liu (2004). This algorithm has two main advantages. First, the number of components is inferred and not assumed to be known

in advance, which is an improvement over previous methods of clustering in networks, for it is seldom the case that one has this information. Moreover, if one is interested in uncovering, rather than validating, some structures in the data, assumptions may not be beneficial. We also do away with additional criteria of model selection, which are generally approximate and thus the not minor problem of assessing their validity is a further complication to be addressed. Second, if the likelihood function is multi-modal, an EM estimate can correspond to one of the many local maxima (see for example (Wu, 1983)), and multiple starting points of the iterations are required to test the optimality of the convergence point. One can argue that Markov chains too can get trapped near local modes when the posterior is multi-modal. While this is true, there are many techniques one can use to monitor the performance of the chains; and, even more importantly, many improvements over standard Monte Carlo methods are now available that allow the sampler to explore the configuration space thoroughly. The Wang-Landau algorithm, which we have implemented, is one such method.

The paper is organized as follows. In Section 2 we review the model. In Section 3 we describe the inference algorithm. In Section 4 we analyze four data sets: one simulated data set, two social data sets, which serve as test-beds for the algorithm, and a biological data set, and report the results. Section 5 concludes and summarizes the paper.

2 Description of the Model

Let $X = (X_{ij})$ be the $N \times N$ symmetric adjacency matrix representing an N-vertex network and encoding many of its topological properties. The parameters of the mixture model for random graphs proposed in (Daudin *et al.*, 2006) are the number of groups, Q; the vector $(\boldsymbol{\alpha}|Q) = (\alpha_1, \ldots, \alpha_Q)$, with α_q being the probability that a given vertex belongs to group q, $q = 1, \ldots, Q$; and the probabilities π_{ql} that there is an edge between vertices belonging to groups q and l, which can be arranged in a $Q \times Q$ matrix called

the connectivity matrix. Introducing the occupation variables

$$Z_{iq} = \mathcal{I}(i \in q),$$

where $\mathcal{I}$ is the indicator function, $i = 1, \ldots N$, and $q = 1, \ldots Q$, one can write

$$\alpha_q = P(Z_{iq} = 1),$$

which implies that the probability that a vertex belongs to a given group does not depend on the vertex itself. Since a vertex must belong to one and only one group, the probabilities $\boldsymbol{\alpha}$ are constrained to sum to one: $\sum_q \alpha_q = 1$. The connectivity matrix, whose elements can be written as

$$\pi_{ql} = P(X_{ij} = 1 | Z_{iq} \cdot Z_{jl} = 1), \qquad q, l = 1, \cdots Q,$$

is symmetric so that one need only consider its $Q(Q+1)/2$ distinct elements. The edges X_{ij} are stochastic variables which are conditionally independent, given the groups to which their vertices belong:

$$X_{ij} | Z_{iq} \cdot Z_{jl} = 1 \sim \mathsf{Bernoulli}(\pi_{ql}). \tag{1}$$

The model is best described by the joint distribution of all variables:

$$P(Q, \boldsymbol{\alpha}, Z, \boldsymbol{\pi}, X) = P(Q)P(\boldsymbol{\alpha}|Q)P(\boldsymbol{\pi}|Q)P(Z|\boldsymbol{\alpha}, Q)P(X|\boldsymbol{\pi}, Z, Q), \tag{2}$$

where some simplifying assumptions have been made:

$$P(\pi|Q, \boldsymbol{\alpha}, Z) = P(\boldsymbol{\pi}|Q),$$

and

$$P(X, Z|Q, \boldsymbol{\alpha}, \boldsymbol{\pi}) = P(Z|Q, \boldsymbol{\alpha}) \cdot P(X|Q, \boldsymbol{\pi}, Z).$$

Under this model, the complete data likelihood is

$$\begin{aligned} L_Q &= P(Z|Q, \boldsymbol{\alpha}) \cdot P(X|Q, \boldsymbol{\pi}, Z) \\ &= \prod_{i=1}^{N} \prod_{q=1}^{Q} \alpha_q^{Z_{iq}} \prod_{1 \le i < j \le N} \prod_{l,q=1}^{Q} \left(\pi_{ql}^{X_{ij}} \cdot (1 - \pi_{ql})^{1 - X_{ij}} \right)^{Z_{iq} \cdot Z_{jl}}. \end{aligned} \tag{3}$$

We use a Markov chain Monte Carlo method to explore the posterior probability of the parameters under model (2). In particular, we are interested in learning about the occupation variables Z_{iq} and the number of groups Q. The use of RJMCMC is most appropriate in this setting, because the parameters depend on the number of groups, which is itself a parameter to be determined. In the next section we give the details of our proposed algorithm.

3 The Method

We want to generate the posterior probabilities under our model in a Bayesian framework. We thus need to specify the prior distributions. The minimum number of groups is 1 and the maximum is N, the number of vertices, but we have no reason to favor one number of groups over another. We thus consider a uniform prior distribution: $P(Q) = 1/N$. We use a Dirichlet prior distribution for $(\alpha_1, \ldots, \alpha_Q)$:

$$P(\boldsymbol{\alpha}|Q) = P(\alpha_1, \ldots, \alpha_Q|a, Q) = \frac{\Gamma(Q \cdot a)}{\Gamma(a)^Q} \prod_{i=1}^{Q} \alpha_i^{a-1} \delta(\sum_i \alpha_i = 1). \tag{4}$$

The probabilities $\boldsymbol{\pi}$ are *a priori* independent for different pairs of groups. We use for each of its $Q \cdot (Q+1)/2$ distinct elements the same Beta prior distribution, so that

$$P(\boldsymbol{\pi}|Q) = \prod_{1 \le l \le q \le Q} P(\pi_{ql}) = \prod_{1 \le l \le q \le Q} \frac{\Gamma(2b)}{\Gamma(b)^2} \pi_{ql}^{b-1} (1 - \pi_{ql})^{b-1}. \tag{5}$$

The Monte Carlo method used in this paper consists of three steps that are iterated until the probability density generated reaches equilibration: two RJMCMC moves (splitting/merging and birth/death) and a Gibbs update. The Gibbs mechanism updates the parameters by drawing them from their full conditional distributions. Therefore,

$$(\alpha_1, \ldots, \alpha_Q)|(Q, \boldsymbol{\pi}^{(k-1)}, Z^{(k-1)}, X) \sim \mathsf{Dirichlet}(a + n_1, \ldots, a + n_Q),$$

where $n_q = \sum_{i=1}^{N} Z_{iq}^{(k-1)}$ is the number of vertices in the group q, and the super-scripts $(k-1)$ and (k) indicate the values of the parameters before and after the Gibbs update;

$$\pi_{qq}|(Q, \boldsymbol{\pi}, Z^{(k-1)}, X, \boldsymbol{\alpha}^{(k)}) \sim \mathsf{Beta}\left(b + \sum_{i<j} X_{ij} Z_{iq} Z_{jq},\ b + \sum_{i<j} (1 - X_{ij}) Z_{iq} Z_{jq}\right),$$

and, for $q \neq l$,

$$\pi_{ql}|(Q, \boldsymbol{\pi}, Z^{(k-1)}, X, \boldsymbol{\alpha}^{(k)}) \sim \mathsf{Beta}\left(b + \sum_{i,j} X_{ij} Z_{iq} Z_{jl},\, b + \sum_{i,j} (1 - X_{ij}) Z_{iq} Z_{jl}\right).$$

The last update of the Gibbs part of the algorithm consists in re-allocating the vertices to the groups one at a time. The vertex i is placed in the cluster q with probability

$$P\left(Z_{iq} = 1 | (Z \setminus Z_{iq})^{(k,k-1)}, Q, \boldsymbol{\pi}^{(k)}, \boldsymbol{\alpha}^{(k)}\right) \propto \alpha_q^{(k)} \prod_{l=1}^{Q} {\pi_{ql}^{(k)}}^{C_{il}} (1 - \pi_{ql}^{(k)})^{\tilde{n}_l - C_{il}}, \qquad (6)$$

where $\sum_j X_{ij} Z_{jl}^{(k,k-1)} = C_{il}$, $\sum_{j \neq i} Z_{jl}^{(k,k-1)} = \tilde{n}_l$, and $(Z \setminus Z_{iq})^{(k,k-1)}$ indicates the occupation variables of all points except for i, some of which have already been re-allocated by the Gibbs sampler.

The Gibbs update is preceded in each iteration by one RJMCMC move, either $Q \to Q + 1$ (splitting one group into two groups) or $Q \to Q - 1$ (merging two groups into one group) with probabilities $R_{Q \to Q+1}$ and $R_{Q \to Q-1} = 1 - R_{Q \to Q+1}$, respectively; and by a birth ($Q \to Q+1$) or a death ($Q \to Q-1$) of an empty group, chosen with the same probabilities. We describe these moves in the context of RJMCMC in the following section.

3.1 A Sketch of RJMCMC

The basic idea of RJMCMC is to consider a process, in which changes of dimension occur, as a Markov chain whose moves can take place among configurations of different dimensions. One requires the transitions to satisfy the detailed balance condition and each move is jointly considered with its reverse move. The dimension parameter in our case is the number of groups and thus a move that changes the dimension is a move that increases or decreases the number of groups. We only consider jumps that change the dimension by one. To fix notation, let us label the generic model with Q components by its parameters, that is by the pair (Q, θ_Q) where the vector θ_Q indicates the Q components $(\alpha_1, \ldots \alpha_Q)$ and the $Q \cdot (Q+1)/2$ distinct elements of the connectivity

matrix $\boldsymbol{\pi}$. When there is no possibility of misinterpretation, we suppress θ. Following the general formalism of (Green, 1995), the move

$$(Q, \theta_Q) \rightarrow (Q+1, \theta_{Q+1}),$$

is accepted with probability $F(A_{Q\rightarrow Q+1})$, where $F(z) = \min\{1, z\}$ is the Metropolis function and

$$A_{Q\rightarrow Q+1} = \frac{L_{Q+1}}{L_Q} \frac{P(\boldsymbol{\pi}|Q+1)P(\boldsymbol{\alpha}|Q+1)}{P(\boldsymbol{\pi}|Q)P(\boldsymbol{\alpha}|Q)} \frac{P(Q+1)}{P(Q)} \frac{R_{Q+1\rightarrow Q}}{R_{Q\rightarrow Q+1}} \frac{Jac_{Q\rightarrow Q+1}}{\Phi(\mathbf{v})}. \tag{7}$$

In the above formula (7), $\mathbf{v}$ is a random vector which has the same dimension as the vector $\theta_{Q+1} - \theta_Q$; $\Phi(\mathbf{v})$ is the density of $\mathbf{v}$; $Jac_{Q\rightarrow Q+1}$ is the Jacobian of the diffeomorphism that maps $(\theta_Q, \mathbf{v})$ to θ_{Q+1}. The probabilities $R_{Q\rightarrow Q\pm1}$ are chosen to be $R_{Q\rightarrow Q\pm1} = 1/2$ for $2 \leq Q \leq N-1$ and $R_{N\rightarrow N-1} = 1 = R_{1\rightarrow 2}$, since we allow at most N groups. The advantage of this formulation is that the reverse move is accepted with probability

$$p_{Q+1\rightarrow Q} = F(1/A_{Q\rightarrow Q+1}).$$

We now define the pairs merging/splitting and birth/death of an empty component for our problem and compute their acceptance probabilities.

Splitting and Merging

Let us assume we are in a configuration with Q groups. We define the splitting, which increases the number of groups by one, $Q \rightarrow Q+1$, as follows. We randomly choose a non-empty group, q, we split it into two groups, and reassign its vertices to either one or the other, independently with probability 1/2. All other $Q-1$ groups are not involved in the move. If $I_{\setminus q}$ is the set of indices of the latter groups, *i.e.* up to permutation $\{q\} \cup I_{\setminus q} = \{1, 2, \ldots, Q\}$, the parameters $\boldsymbol{\alpha}$ change as follows

$$\begin{aligned} (\boldsymbol{\alpha}|Q) &\rightarrow (\tilde{\boldsymbol{\alpha}}|Q+1), \\ (\alpha_{I_{\setminus q}}, \alpha_q) &\rightarrow (\alpha_{I_{\setminus q}}, \alpha_q \cdot p, \alpha_q \cdot (1-p)), \end{aligned} \tag{8}$$

where we choose to draw p from a Beta distribution $\mathsf{Beta}(c, c)$, whose parameter c is fixed for all splitting moves in the Markov chain. Of the $(Q+1)\cdot(Q+2)/2$ distinct elements of the connectivity matrix in the new configuration, only those involving the two new groups need to be redefined. Let us label the two new groups q_1 and q_2, and define the transformation

$$(\boldsymbol{\pi}|Q) \to (\tilde{\boldsymbol{\pi}}|Q+1) \tag{9}$$

as follows

$$\begin{aligned}
\tilde{\pi}_{i\,q_1} &= \frac{\pi_{iq} - p_i}{1-p_i}, \qquad i \in I_{\setminus q}, \\
\tilde{\pi}_{i\,q_2} &= p_i \qquad i \in I_{\setminus q}, \\
\tilde{\pi}_{q_2\,q_2} &= p_{Q+1}, \\
\tilde{\pi}_{q_1 q_1} &= p_q, \\
\tilde{\pi}_{q_1\,q_2} &= \frac{\pi_{qq} - (p_q + p_{Q+1} - p_{Q+1}\cdot p_q)}{(1-p_q)(1-p_{Q+1})}, \\
\tilde{\pi}_{kl} &= \pi_{kl}, \qquad k,l \in I_{\setminus q}, \qquad k<l,
\end{aligned}$$

where the $Q+1$ numbers $(p_1, \ldots, p_{Q+1})$ are drawn from the following distributions: p_i from the uniform distribution $U[0, \pi_{iq}]$, for $i = 1, \ldots, q, \ldots, Q$, and p_{Q+1} from $U[0, (\pi_{qq} - p_q)/(1-p_q)]$, to ensure $\tilde{\pi} \in [0,1]$. For the reverse move $Q+1 \to Q$, we randomly choose two groups, q and l, one of which at least non-empty, and merge them into a group $\tilde{l}$. The parameters in the new configuration are

$$(\boldsymbol{\alpha}|Q+1) = (\alpha_l, \alpha_q, \alpha_{I_{\setminus(q,l)}}) \to (\tilde{\boldsymbol{\alpha}}|Q) = (\tilde{\alpha}_{\tilde{l}}, \alpha_{I_{\setminus(q,l)}}),$$

where $\tilde{\alpha}_{\tilde{l}} = \alpha_l + \alpha_q$, and

$$\begin{aligned}
\tilde{\pi}_{i\tilde{l}} &= \pi_{il} + \pi_{iq} - \pi_{il}\cdot\pi_{iq}, \qquad i \in I_{\setminus(q,l)}, \\
\tilde{\pi}_{\tilde{l}\tilde{l}} &= \pi_{ll} + \pi_{qq} + \pi_{lq} + \pi_{ll}\cdot\pi_{qq}\cdot\pi_{lq} - \pi_{ll}\cdot\pi_{qq} - \pi_{qq}\cdot\pi_{lq} - \pi_{ll}\cdot\pi_{lq}.
\end{aligned}$$

The occupation number of the new group $\tilde{l}$ is

$$\sum_{i=1}^{N} Z_{i\tilde{l}} = \tilde{n}_{\tilde{l}} = n_l + n_q.$$

With the above definitions, formula (7) becomes

$$\begin{aligned} A_{Q\to Q+1}(\theta_Q,\theta_{Q+1}) &= \frac{L_{Q+1}}{L_Q}\frac{P(\boldsymbol{\pi}|Q+1)P(\boldsymbol{\alpha}|Q+1)}{P(\boldsymbol{\pi}|Q)P(\boldsymbol{\alpha}|Q)}\cdot\nu\cdot \\ &\quad \alpha_q\cdot\prod_{i=1}^{Q+1}\frac{1}{1-p_i}\cdot\frac{(\pi_{qq}-p_q)\prod_{i=1}^{Q}\pi_{iq}}{\frac{\Gamma(2c)}{\Gamma(c)^2}p^{c-1}(1-p)^{c-1}\cdot(1-p_q)}, \end{aligned} \tag{10}$$

where

$$\nu=\frac{2^{n_q}\cdot 3\cdot 2^{Q+1}\cdot(Q-Q_0)}{Q(Q+1)-((Q+1)_0-1)\cdot(Q+1)_0}\frac{R_{Q+1\to Q}}{R_{Q\to Q+1}},$$

and Q_0 is the number of empty groups in the configuration with Q groups.

Birth and Death of an Empty Component

The death and birth of an empty component are generally implemented in RJMCMC algorithms to improve mixing. The birth of an empty component increases the number of groups by one: $Q\to Q+1$. The parameters are modified in the following way:

$$\begin{aligned} (\alpha_1,\ldots,\alpha_q,\ldots,\alpha_Q) &\to ((1-\lambda)\cdot\alpha_1,\ldots,(1-\lambda)\cdot\alpha_Q,\lambda), \\ (\boldsymbol{\alpha}|Q) &\to (\tilde{\boldsymbol{\alpha}}|Q+1), \end{aligned} \tag{11}$$

where λ is drawn from a Beta distribution $\mathsf{Beta}(d,Q\cdot d)$. The additional $Q+1$ numbers p_i, $i=1,\ldots,Q+1$, that are needed to describe the edge probabilities among the new group and the others, are drawn from the Beta distribution $\mathsf{Beta}(b,b)$. For the reverse move (death), $Q+1\to Q$, one randomly selects one of the Q_0+1 empty components and eliminates it. The parameters π_{ql}, for $l=1,\cdots,Q+1$, are suppressed, as is α_q, where q is the label of the empty component. The α of all other components are rescaled to sum to one:

$$(\alpha_1,\ldots,\alpha_q,\ldots,\alpha_{Q+1})\to(\alpha_1/(1-\alpha_q),\ldots,\hat{\alpha_q},\ldots,\alpha_{Q+1}/(1-\alpha_q)),$$

where the caret indicates that the corresponding value is missing. For the birth/death just described, formula (7) becomes

$$A_{Q\to Q+1}(\theta_Q,\theta_{Q+1}) = (1-\lambda)^N\frac{1}{Q_0+1}\frac{R_{Q+1\to Q}}{R_{Q\to Q+1}}\cdot\nu, \tag{12}$$

where

$$\nu = \frac{\Gamma((Q+1)\cdot a)\Gamma(Q\cdot d)\Gamma(d)}{\Gamma((Q+1)\cdot d)\Gamma(Q\cdot a)\Gamma(a)}\lambda^{a-d}(1-\lambda)^{Q(a-d)}.$$

3.2 Improving Acceptance and Rejection Rates

It is hard to engineer reversible jump Markov chains that enjoy high acceptance and rejection rates. To improve the efficiency of the sampler, we modify the acceptance and rejection probabilities, that is equation (7), in the way described by Atchadé and Liu (2004), who generalized the Wang-Landau algorithm (Wang and Landau, 2001) to general state space. In order to visit all possible configurations, the groups most often visited by the sampler are penalized. To each configuration i, $i = 1, \dots N$, a weight $\phi(i)$ is assigned that is updated after each splitting/merging move in the following way

$$\phi_k(i) = \phi_{k-1}(i) \cdot (1 + \gamma_k \cdot \mathcal{I}(i = Q)),$$

where Q is the number of groups of the configuration the sampler is visiting and k labels the iteration step. Rescaling the acceptance probabilities of a splitting move

$$p(Q \to Q+1) = F\left(A_{Q\to Q+1} \cdot \frac{\phi_k(Q)}{\phi_k(Q+1)}\right),$$

and of a merging move

$$p(Q+1 \to Q) = F\left(\frac{1}{A_{Q\to Q+1}} \cdot \frac{\phi_k(Q+1)}{\phi_k(Q)}\right),$$

where $A_{Q\to Q+1}$ is given by (7), has the effect of favoring transitions toward states that are less visited. The subtle part of the algorithm is the update of γ_n. Each time all possible groups have been visited approximately the same number of times, as indicated by the flatness of frequency histograms, γ is updated according to the following schedule:

$$\gamma_{i+1} + 1 = f(\gamma_i + 1) = (\gamma_i + 1)^{1/2},$$

where γ_i is the value at the i-th update and γ_{i+1} is the updated value. The square root function can in fact be substituted by any function f that lets γ decrease monotonically to zero. The values of γ can be used to check convergence, as it is enough to ensure

that γ_n is close to zero, *i.e.* smaller than a chosen very small value. We use $\gamma_0 = \exp(1)$, $\phi_0(i) = 1$ for all i, and, as a criterion of flatness for the frequency histogram, that the minimum frequency should be at least one fourth of the average frequency. Since, in general, only a small fraction of the groups are non-empty, in checking the flatness of the histograms, we evaluate the average and the minimum in the interval $[\max(1, (Q - Q_0)_{min} - \epsilon), \min(N, (Q - Q_0)_{max} + \epsilon)]$, rather than in the interval $[1, N]$, where $(Q-Q_0)_{min}$ and $(Q-Q_0)_{max}$ are the minimum and maximum number of occupied groups, each computed anew every time the histogram is flat. Notice that flatness is monitored over an interval of length greater than ϵ, so that a choice of $\epsilon = N/3$, which is not too drastic, makes the convergence quicker especially when the possible maximum number of groups N is large.

4 Applications

We present here the results of the application of the method to four networks. Specifically, we ran two versions of the algorithm: one with and the other without the Wang-Landau extension, to which we refer as RJMCMC and RJMCMCWL respectively. The results obtained are compatible. By sweeping through all possible values of Q, RJMCMCWL is more appealing in situations where the probability density has many local modes and thus the likelihood of not visiting a sufficiently large part of the configuration space is considerable. However, it is computationally more expensive.

To represent the identified clusters, we construct a co-membership network, whose vertices are the vertices of the original network, and whose edges are labeled with weights that measure the co-membership frequency of pairs of vertices in the samples; that is, the weight of edge (i, j) counts the proportion of samples in which vertex i and vertex j are in the same component. We notice that these quantities, pair-wise posterior probabilities that two vertices are in the same component, are invariant under permutation of the labels of the mixture components. If instead we were explicitly interested in the component parameters, we would have to undo the label switching in the MCMC out-

put (Jasra *et al.*, 2005), by relabeling the output so as to make the marginal posterior distributions of the parameters of interest unimodal, at least to a certain extent.

We actually fix a threshold value τ, so that an edge of the co-membership network is drawn if and only if its weight is greater than or equal to τ.

For brevity, the analysis of the sensitivity of the results to the choice of prior distributions is reported in some detail only for the simulated network and one of the real networks, although we used different hyper-parameters in all data sets. We also considered different initial conditions: random assignments of vertices and initial number of groups.

4.1 A Simulation Study

We simulated a network with $Q = 5$ groups and $N = 60$ vertices. Each vertex was independently assigned to one of the groups using a multinomial distribution with probabilities $\boldsymbol{\alpha}$ sampled from a Dirichlet distribution. The clusters thus obtained are written in Table 1. Formula (1), with π_{ql} samples from a Beta distribution, was used to decide whether a pair of vertices should be joined by an edge; the resulting network, which has 497 edges, is displayed in Figure 1. Table 2 shows the inferred probabilities $P(Q - Q_0)$ of the number of non-empty groups for some choices of hyper-parameters. There is some sensitivity to the values of the prior distributions: when b is very different from 1, the algorithm tends to merge different groups, with the more connected groups coalescing first; when a is very different from 1, the one-vertex group (the third in Table 1) merges with the fifth. Figure 2 shows the co-membership network for almost uniform prior distributions, using a threshold $\tau = 0.7$. The solution is stable over a reasonable interval $\Delta\tau$; namely an identical network is found at $\tau = 0.5$, and at $\tau = 0.8$ the point 37 stands isolate. In Figure 3, we report the histograms for both $P(Q)$ and $P(Q - Q_0)$ obtained using RJMCMCWL for one of the combinations of hyper-parameters with which we experimented. The histogram for $P(Q)$ is quite flat, as one should expect, and the distribution of the number of occupied groups is compatible with and a bit more spread-out

than that obtained by RJMCMC with the same hyper-parameters (see Table 2).

4.2 Analysis of Real Networks

The first real network we analyze is known as the (unweighted) karate club network (Zachary, 1977). It describes the social interactions of members of a karate club, who after some internal dispute broke up into two groups, depicted with two different colors in Figure 4. The results obtained using the RJMCMC algorithm show some dependence on the values of the hyper-parameters. Table 3 summarizes the inferred $P(Q)$ and Table 4 shows the inferred $P(Q-Q_0)$ for some choices of hyper-parameters. For uniform prior distributions, *i.e.* $a = b = c = d = 1$, the inferred occupation variables Z_{iq} are represented in Figure 5. Notice that in a co-membership network the expected solution that groups the vertices $1-8, 11-14, 17, 18, 20, 22$ and the vertices $9, 10, 15, 16, 19, 21, 23-34$ corresponds to two connected components, each being a clique, *viz.* any two points being connected by a vertex. One can see that our result is quite accurate and compatible with the solution. Vertices representing people who belong to different groups after the dispute are never clustered together by the algorithm, as expected. Even better, the algorithm is able to identify some interesting sub-structures: *e.g.* vertex 33 and vertex 34 form their own cluster owing to their very similar edge structure in the original social network (Figure 4); vertex 1 is singled out because it is linked to all vertices but one belonging to the same group to which it belongs after the dispute. Varying the values of the hyper-parameters c and d does not change this result. However, for large values of b, for $a < 1$, and $b < 1$ the algorithm tends to classify all points as belonging to one group or to coalesce the two main groups of Figure 5. RJMCMCWL has a sensitivity to the choice of prior distributions similar to that of RJMCMC, for it is the same Gibbs mechanism that assigns vertices to groups at each sweep in both versions of the algorithm. Table 5 shows the inferred probability $P(Q-Q_0)$, which is generally a bit more spread-out and with slightly higher median than the corresponding probability distribution obtained with RJMCMC (Table 4). The inferred probabilities $P(Q)$, which are not reported here, are, on the contrary, quite flat, as they should be. Runs were stopped

when $\gamma \approx 0.001$ at least, a value which required a number of iterations that depended on the prior distribution chosen. In particular, for $a = b = c = d = 1.1$, 12 million iterations were necessary to reach $\gamma = 0.0009$. With this choice of hyper-parameters, the proportion of combined merging/splitting moves accepted was of 40 per cent and that of death/birth of 50 per cent. The predicted modules are shown in Figure 6.

We now turn to a network that describes the social interactions of a school of 62 dolphins (Lusseau *et al.*, 2003; Lusseau, 2003), which was downloaded from the web-site http://www-personal.umich.edu/$\sim$ mejn/netdata. The network vertices represent the dolphins and the 159 edges represent their companionship. The problem of identifying communities in this network was studied in (Lusseau and Newman, 2004) using the betweenness-based algorithm of Girvan and Newman (GN) (2002). Figure 7 shows the results of the analysis carried out using the RJMCMC algorithm and the RJMCMCWL algorithm, respectively, with the same values of hyper-parameters $(a, b, c, d) = (1, 10, 1, 1)$. The two colors depict the two communities identified by the GN algorithm. The RJMCMC identifies the same two communities as the GN algorithm except for two vertices. One of these vertices, number 40 in Figure 7, which represents dolphin SN89, has degree two in the dolphin companionship network and its two neighbors belong to the two big clusters in Figure 7(b). Furthermore, if we use the lower threshold $\tau = 0.5$, the co-membership network (not reported here) has only one connected component with SN89 bridging between two clearly defined sub-components which are otherwise cliques: namely, if we removed the vertex SN89, the co-membership network at $\tau = 0.5$ would be the same as that at $\tau = 0.8$, which is Figure 7(b). It thus seems that SN89 can be assigned to neither of the two large communities. This is a nice feature of the algorithm. We are not fixing the number of clusters and thus the vertices are not forced to belong to one of the clusters. A vertex can be its own cluster, either because it has a special status, as the application of the algorithm to the Zachary's network seems to suggest or because there may be more than one community to which it can belong, *i.e.* it is unclassified, as this application seems to hint at. The fact that τ is a probability can help the interpretation of the results. The RJMCMCWL

solution is more fragmented, which is tantamount to the fact that the posterior probabilities $P(Q - Q_0)$ are more spread. The GN algorithm identifies 3 sub-communities in the largest community (with white nodes in Fig. 7). Three of the groups identified by RJMCMCWL (the triangle, the 6-vertex module and the linear 4-vertex module) are each made up with vertices belonging to one of these sub-communities. Uniform prior distributions fragment the modules a bit more. We do not know the true clusters in this network, so it may well be that our algorithm identifies some sub-communities that the GN algorithm is unable to discover.

Finally, we apply the algorithm to the network of gap junctions in the C. Elegans. The junctions are the edges of the network and the neurons are the nodes. The specific network we use is that considered in (Chen *et al.*, 2006) and (Varadan *et al.*, 2006), which can be found at the web-site http://www.ee.columbia.edu/anastas/ismsb2006. We limit our analysis to the largest connected component of the (280-vertex) complete network, which has 248 nodes and 511 edges, after the elimination of 3 loops. Using almost uniform prior distributions, $a = b = c = d = 1.1$, and a threshold $\tau = 0.8$, the co-membership network contains 7 components with more than 5 nodes. We are aware of no papers where the identification of modules in this network has been studied and to which we can compare the results found here. However, following (Varadan *et al.*, 2006), we can use some information on gene expression profiles on the neurons to help us understand whether the modules found by the algorithm may be plausible. Table 6 summarizes the genes that are expressed in more than 50 percent of the neurons in the four largest modules and the module sizes. The two smallest modules are very homogeneous in the sense that there are a few genes that are expressed in almost the totality of the neurons. Of course, what the algorithm is programmed to detect are clusters of vertices in the network of gap junctions, which *a priori* has nothing to do with the genetics of the C. Elegans. However, it is believed (Varadan *et al.*, 2006) that some genes may have an effect on the creation of gap junctions between neurons. It may just be a coincidence rather than an indirect confirmation of the validity of the algorithm, but we find it interesting that the smallest modules have a genetic homogeneity.

5 Conclusions

In this paper we have considered the important problem of clustering vertices in networks. We have used mixture distributions to model the heterogeneity of the vertices. We have proposed a reversible jump Markov chain Monte Carlo algorithm to infer the model parameters including the number of clusters, which is an improvement over the algorithms used in the very few papers where finite mixtures have been employed in graphs (Daudin *et al.*, 2006; Newman and Leicht, 2007), where the number of clusters is fixed or at best determined in an indirect way. Even the Bayesian method proposed in (Handcock *et al.*, 2007) in the very different theoretical framework of latent structure models resorts to a Bayesian information criterion approximation to select the number of clusters. Moreover, we have implemented the Wang-Landau algorithm to improve sampling over standard Markov chains.

We have tested the algorithm on 3 real-data sets: one social data set, where the module structure is known; a second data set, where a similar investigation was carried out using a different algorithm; and a third biological data set. We have also analyzed a simulated network. The results are encouraging.

The method and the model have some limitations. We highlight two of them. In its present formulation, the model can only be used for undirected network. However, a modification of the model that allows for directed networks appears to be feasible. The second limitation is that the algorithm is computationally intensive, especially in the re-weighted version, namely where a variant of the WL algorithm is considered. In this latter case, the algorithm becomes impracticable for very large networks. There might exist better definitions of the splitting/merging moves that could improve the computational performance of the algorithm even in the absence of re-weighting. Integrating the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\pi}$ out in the acceptance probabilities will speed up the algorithm and it would seem quite natural to do so, especially if one limits one's analysis to infer the occupation variables Z_{iq} only, as we have done.

Acknowledgments

We acknowledge the support of the NIH under grant CA127334. We should like to thank Mahlet G. Tadesse for discussions, the editors and two referees for their suggestions.

References

Albert, R. and Barabási, A.B. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, **74,** 47–97.

Atchadé, Y. and Liu, J. (2004). The Wang-Landau algorithm for Monte Carlo computation in general state space. Technical Report. University of Ottawa. http://www.mathstat.uottawa.ca/∼yatch436/gwl.pdf.

Barabási, A.B. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, **286,** 509–512.

Chen, B.I., Hall, D.H. and Chklovskii, D.B. (2006). Wiring optimization can relate neural structure and function. *Proc. Natl. Acad. Sci. USA*, **103,** 4723–4728.

Danon, L., Díaz-Guilera, A., Duch, J. and Arenas, A. (2005). Comparing community structure identification. *Journal of Stat. Mech.*, P09008.

Daudin, J.J., Picard, F. and Robin, S. (2006). A mixture model for random graphs. Technical Report. INRIA. Report Number 5840. http://www.inria.fr/rrrt/rr-5840.html.

Erdös, P. and Rényi, A. (1959). On random graphs I. *Publ. Math.*, **6,** 290–297.

Gilbert, E.N. (1959). Random graphs. *Ann. Math. Stat.*, **30,** 1141–1144.

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

Green, P.J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, **82,** 711–732.

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

Jasra, A., Holmes, C.C. and Stephens, D.A. (2005). Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modeling. *Statistical Science*, **20,** 50–67.

Lusseau, D. (2003). The emergent properties of a dolphin social network. *Proc. R. Soc. London* B, **270,** S186–S188.

Lusseau, D. and Newman, M.E. (2004). Identifying the role that animals play in their social networks. *Proc. R. Soc. London B (suppl.)*, **271,** S477–S481.

Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E. and Dawson, S. M. (2003). The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, **54,** 396–405.

Newman, M.E.J. and Leicht, E.A. (2007). Mixture models and exploratory data analysis in networks. *Proc. Natl. Acad. Sci. USA*, **104,** 9564–9569.

Titterington, D.M., Smith, A.F.M. and Makov, U.E. (1985). *Statistical analysis of finite mixture distributions*. New York: Wiley.

Varadan, V., Miller, D.M. and Anastassiou, D. (2006). Computational inference of the molecular logic for synaptic connectivity in *C. Elegans*. *Bioinformatics*, **22,** 497–506.

Wang, F. and Landau, D.P. (2001). An efficient, multiple range random walk algorithm to calculate the density of states. *Phys. Rev. Lett.*, **86,** 2050.

Watts, D.J. and Strogatz, S.H. (1998). Collective dynamics of "small-world" networks. *Nature*, **393,** 440–442.

Wu, C.F. Jeff (1983). On the convergence properties of the EM algorithm. *Ann. Stat.*, **11,** 95–103.

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

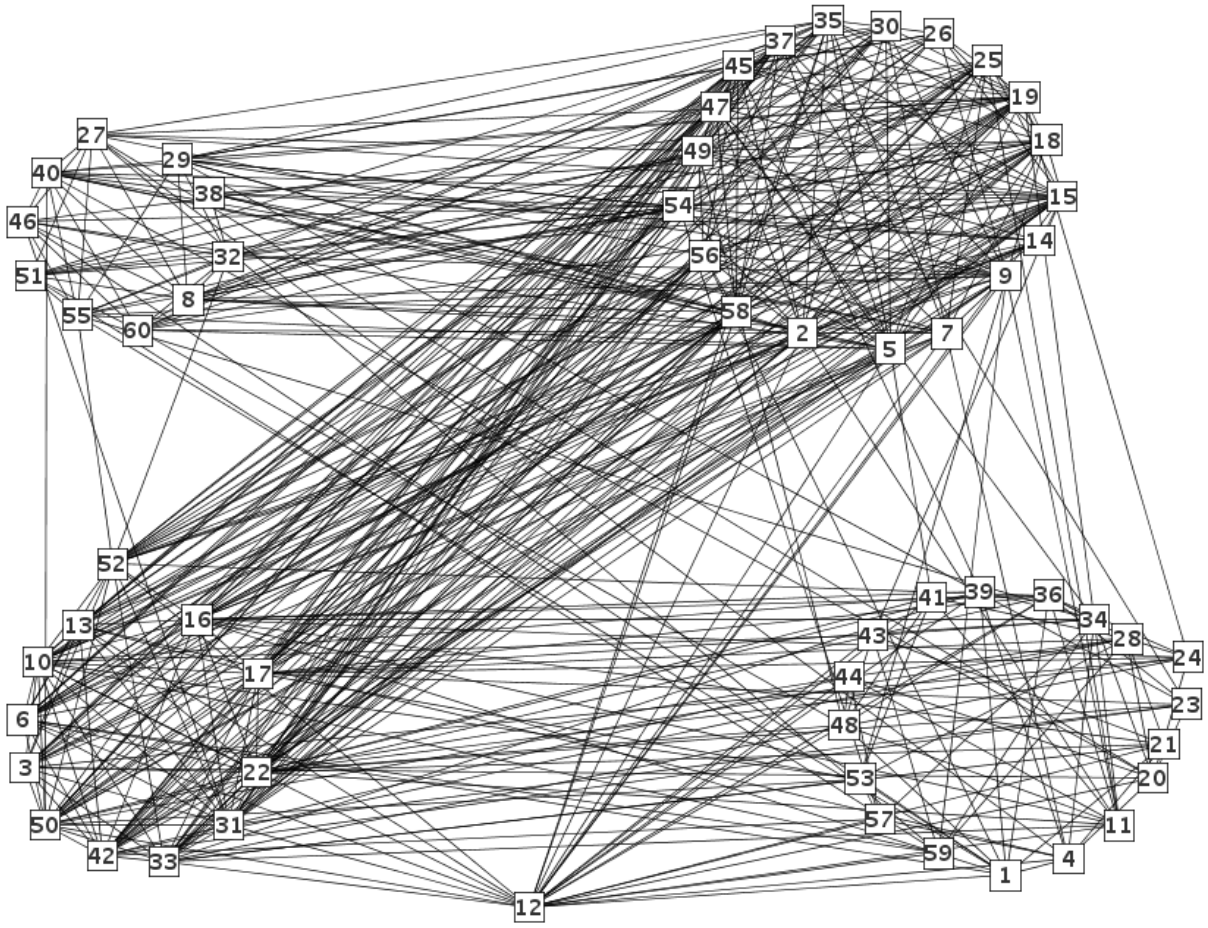


Figure 1: The simulated network.

Cluster	Nodes
1	8, 27, 29, 32, 38, 40, 46, 51, 55, 60
2	2, 5, 7, 9, 14, 15, 18, 19, 25, 26, 30, 35, 37, 45, 47, 49, 54, 56, 58
3	12
4	1, 4, 11, 20, 21, 23, 24, 28, 34, 36, 39, 41, 43, 44, 48, 53, 57, 59
5	3 6, 10, 13, 16, 17, 22, 31, 33, 42, 50, 52

Table 1: Cluster structure of the simulated network.

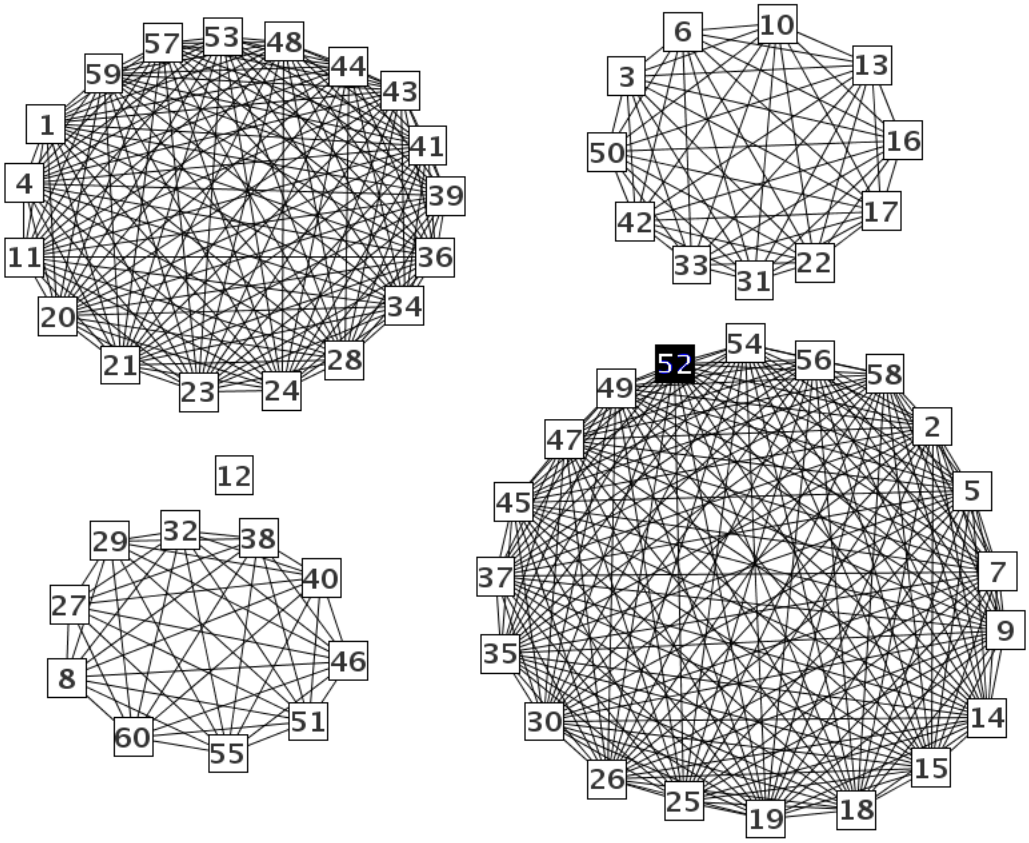


Figure 2: Network representation for the inferred modules in the simulated network. The results are obtained with the RJMCMC algorithm using the hyper-parameters $(a, b, c, d) = (1.1, 1.1, 1.1, 1.1)$. A link between two vertices indicates that the vertices are in the same component in at least 70 percent of the samples. The vertex in black is misclassified by the algorithm.

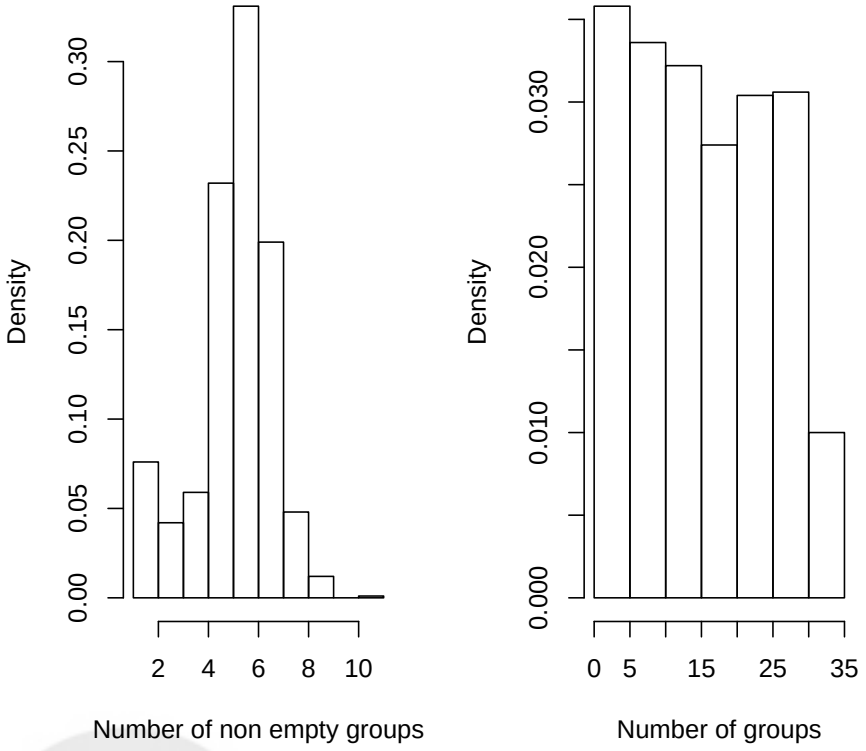


Figure 3: Histograms for the inferred probabilities $P(Q - Q_0)$ (left) and $P(Q)$ (right) obtained with the RJMCMCWL algorithm using the hyper-parameters $(a, b, c, d) = (1.1, 10, 1.1, 1.1)$.

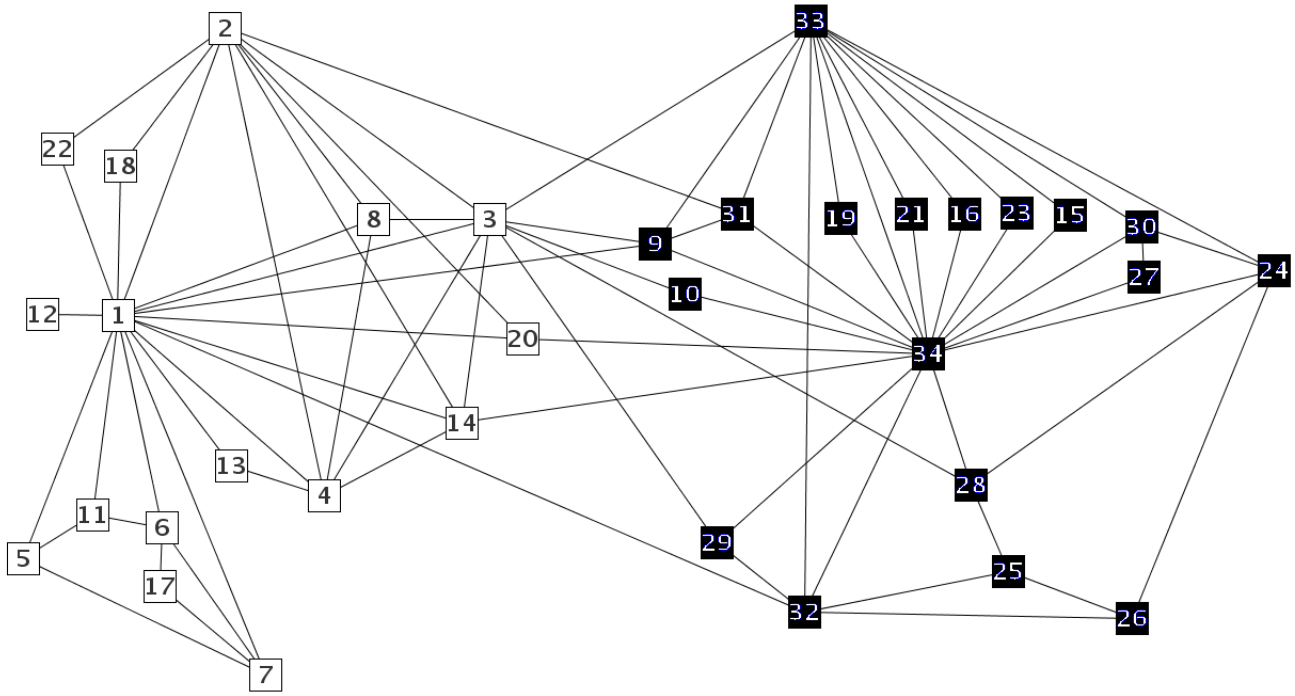
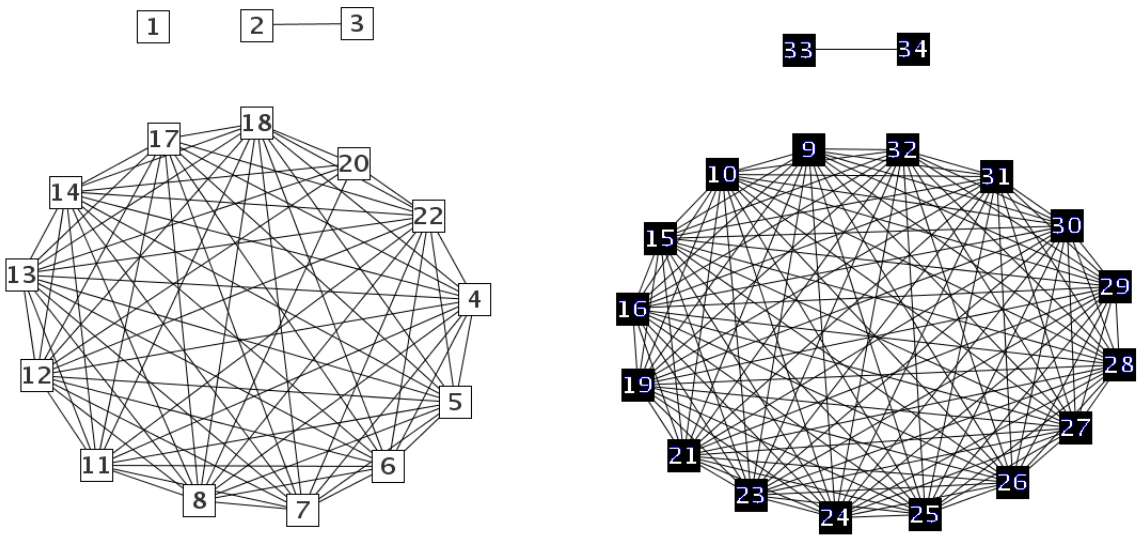
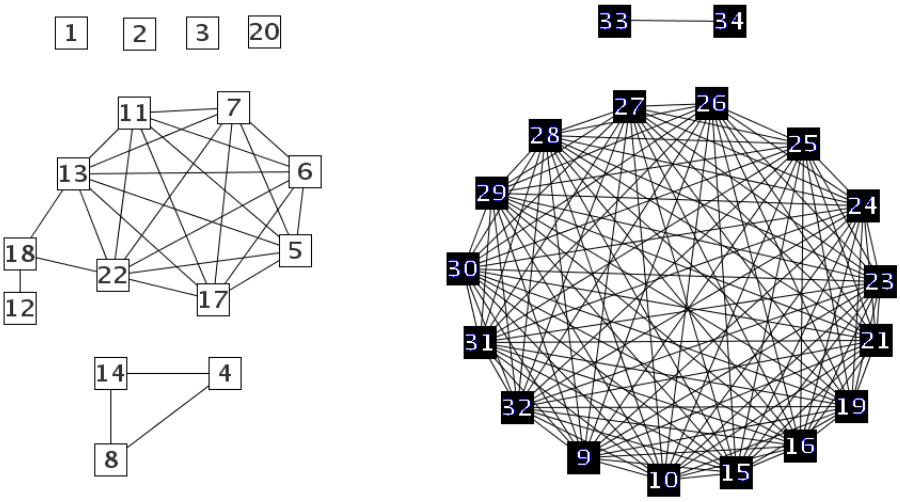


Figure 4: Zachary's karate club network.

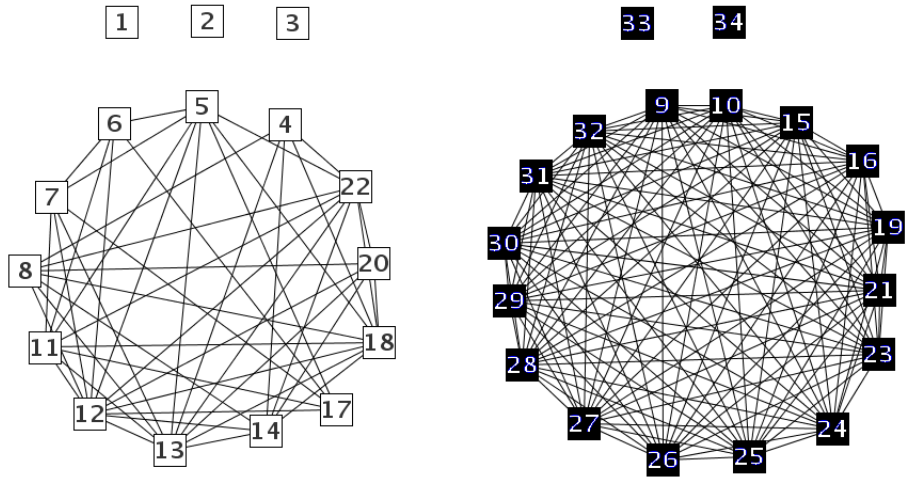


(a) $\tau = 0.5$

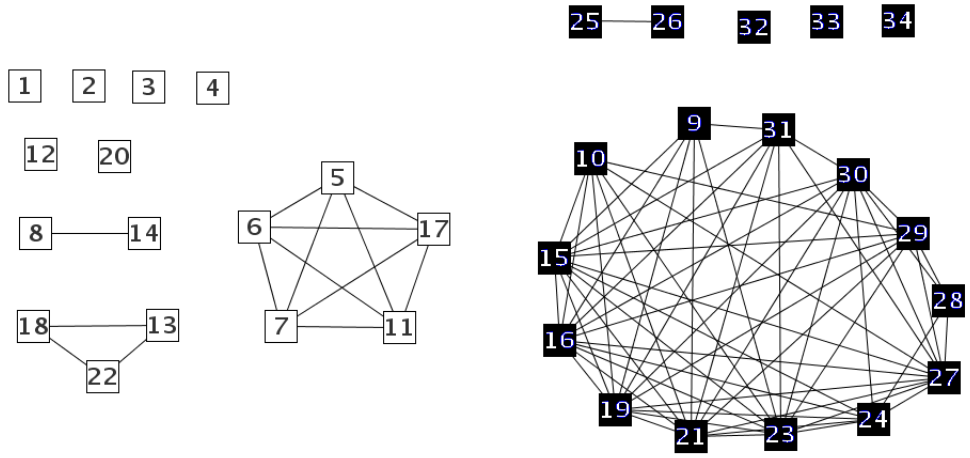


(b) $\tau = 0.8$

Figure 5: Network representation for the inferred modules in Zachary's karate club network. The results are obtained with the RJMCMC algorithm and hyper-parameters $(a, b, c, d) = (1, 1, 1, 1)$. A link between two vertices indicates that the vertices are in the same component in at least (a) 50 percent and (b) 80 percent of the samples.

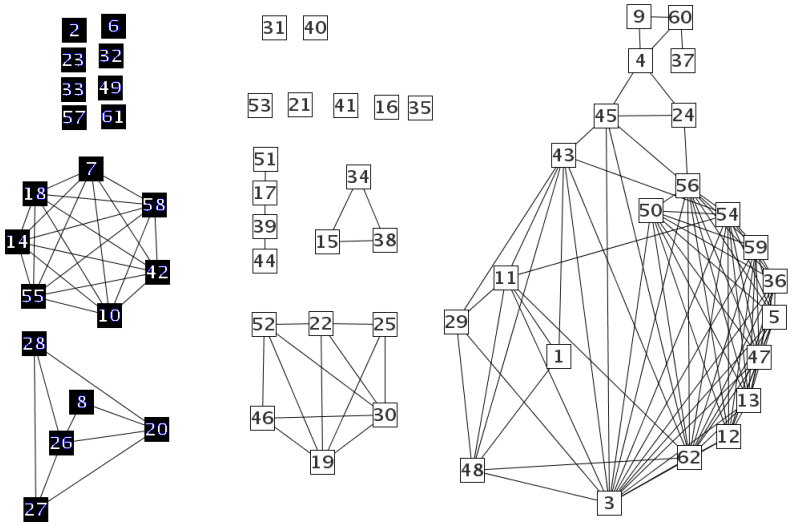


(a) $\tau = 0.5$

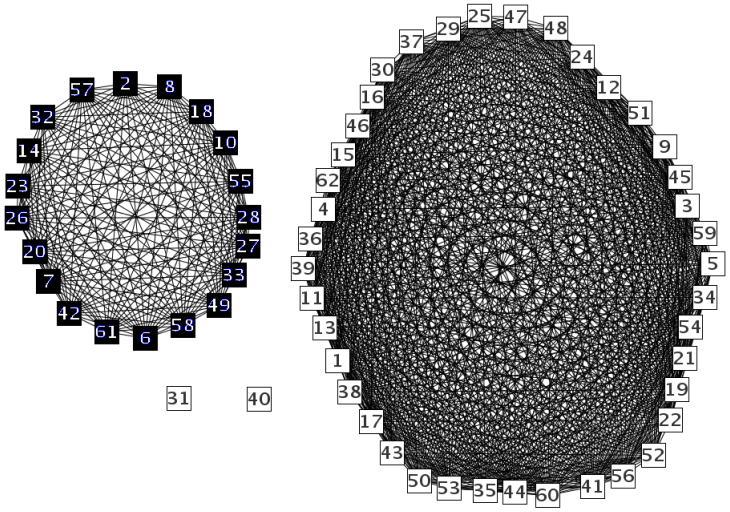


(b) $\tau = 0.8$

Figure 6: Network representation for the inferred modules in the Zachary's karate club social network. The results are obtained with the RJMCMCWL algorithm using the hyper-parameters $(a, b, c, d) = (1.1, 1.1, 1.1, 1.1)$. A link between two vertices indicates that the vertices are in the same component in at least (a) 50 percent and (b) 80 percent of the samples.



(a) RJMCMCWL $\tau = 0.8$



(b) RJMCMC $\tau = 0.8$

Figure 7: Network representation for the inferred modules in the dolphin social network. The results are obtained with (a) the RJMCMCWL algorithm and (b) the RJMCMC using the same hyper-parameters $(a, b, c, d) = (1, 10, 1, 1)$. A link between vertices indicates that the vertices are in the same component in at least 80 percent of the samples. The two colors identify the clusters obtained by the Girvan-Newman algorithm.

Hyper-parameters of the prior distributions (a, b, c, d)	Number of non-empty groups $Q - Q_0$					
	1	2	3	4	5	6
(1.1, 1.1, 1.1, 1.1)	0	0	0	0.08	0.86	0.06
(10, 1.1, 1.1, 1.1, 1.1)	0	0	0	0.66	0.29	0.05
(0.1, 1.1, 1.1, 1.1)	0	0	0	0.14	0.82	0.04
(0.001, 1.1, 1.1, 1.1)	0	0	0	0.89	0.11	0
(1.1, 1.1, 10, 10)	0	0	0	0.04	0.92	0.04
(1.1, 1.1, 0.001, 0.001)	0	0	0	0.09	0.84	0.07
(1.1, 10, 1.1, 1.1, 1.1)	0	0	0	0.88	0.12	0
(1.1, 0.1, 1.1, 1.1)	0.55	0.42	0.03	0	0	0
(1.1, 0.001, 1.1, 1.1)	0.99	0.01	0	0	0	0

Table 2: Inferred posterior probabilities $P(Q - Q_0)$ for the simulated network. Results obtained using RJMCMC. a is the Dirichlet hyper-parameter of the component weights, $\boldsymbol{\alpha}$; b is the Beta hyper-parameter of the probabilities of edges between components, $\boldsymbol{\pi}$, and is also the parameter of the Beta distribution from which the probabilities of edges with a vertex in the group created by a birth move are sampled; c and d are the parameters of the Beta distributions from which random variables are drawn for the reallocation of $\boldsymbol{\alpha}$ in the splitting and birth moves respectively.

Hyper-parameters of the prior distributions	Number of groups Q							
(a, b, c, d)	1	2	3	4	5	6	7	8
(1.2, 10, 1.2, 1.2)	0	0.735	0.215	0.043	0.007	0	0	0
(1.2, 0.001, 1.2, 1.2)	0.98	0.02	0	0	0	0	0	0
(1.2, 1.2, 10, 1.2)	0	0.001	0.004	0.277	0.522	0.170	0.026	0
(1.2, 1.2, 1.2, 10)	0	0.002	0.003	0.249	0.560	0.164	0.021	0.001
(10, 1.2, 1.2, 1.2)	0	0	0.008	0.590	0.367	0.033	0.002	0
(0.001, 1.2, 1.2, 1.2)	0	0.345	0.432	0.158	0.055	0.008	0.002	0
(1.2, 1.2, 10, 10)	0	0	0.005	0.255	0.548	0.145	0.044	0.003
(1.2, 1.2, 0.001, 0.001)	0	0.003	0.005	0.283	0.543	0.144	0.020	0.002
(1, 1, 1, 1)	0	0	0	0.167	0.548	0.259	0.023	0.003

Table 3: Inferred posterior probabilities $P(Q)$ in Zachary's karate club network. These results were obtained using RJMCMC. a is the Dirichlet hyper-parameter of the component weights, $\boldsymbol{\alpha}$; b is the Beta hyper-parameter of the probabilities of edges between components, $\boldsymbol{\pi}$, and is also the parameter of the Beta distribution from which the probabilities of edges with a vertex in the group created by a birth move are sampled; c and d are the parameters of the Beta distributions from which random variables are drawn for the reallocation of $\boldsymbol{\alpha}$ in the splitting and birth moves respectively.

Hyper-parameters of the prior distributions	Number of non-empty groups $Q - Q_0$							
(a, b, c, d)	1	2	3	4	5	6	7	8
$(1.2, 10, 1.2, 1.2)$	0	0.802	0.190	0.008	0	0	0	0
$(1.2, 0.001, 1.2, 1.2)$	1	0	0	0	0	0	0	0
$(1.2, 1.2, 10, 1.2)$	0	0.001	0.004	0.324	0.600	0.071	0	0
$(1.2, 1.2, 1.2, 10)$	0	0.001	0.003	0.309	0.614	0.072	0.001	0
$(10, 1.2, 1.2, 1.2)$	0	0	0.007	0.592	0.373	0.025	0.003	0
$(0.001, 1.2, 1.2, 1.2)$	0	0.985	0.007	0.008	0	0	0	0
$(1.2, 1.2, 10, 10)$	0	0	0.008	0.308	0.608	0.072	0.004	0
$(1.2, 1.2, 0.001, 0.001)$	0	0.003	0.005	0.340	0.580	0.069	0.003	0
(1, 1, 1, 1)	0	0	0	0.219	0.655	0.119	0.007	0

Table 4: Inferred posterior probabilities $P(Q - Q_0)$ in Zachary's karate club network. Results obtained using RJMCMC. a is the Dirichlet hyper-parameter of the component weights, $\boldsymbol{\alpha}$; b is the Beta hyper-parameter of the probabilities of edges between components, $\boldsymbol{\pi}$, and is also the parameter of the Beta distribution from which the probabilities of edges with a vertex in the group created by a birth move are sampled; c and d are the parameters of the Beta distributions from which random variables are drawn for the reallocation of $\boldsymbol{\alpha}$ in the splitting and birth moves respectively.

Hyper-parameters of the prior distributions (a, b, c, d)	Number of non-empty groups $Q - Q_0$						
	1	2	3	4	5	6	
$(10, 1.1, 1.1, 1.1)$	0.019	0.021	0.027	0.029	0.021	0.035	
$(0.001, 1.1, 1.1, 1.1)$	0.038	0.631	0.061	0.205	0.065	0	
$(1.1, 1.1, 1.1, 1.1)$	0.035	0.038	0.030	0.023	0.044	0.120	
$(1.1, 10, 1.1, 1.1)$	0.038	0.057	0.126	0.256	0.352	0.153	
	7	8	9	10	11	12	13
$(10, 1.1, 1.1, 1.1)$	0.093	0.232	0.277	0.169	0.059	0.016	0.002
$(0.001, 1.1, 1.1, 1.1)$	0	0	0	0	0	0	0
$(1.1, 1.1, 1.1, 1.1)$	0.240	0.270	0.173	0.027	0	0	0
$(1.1, 10, 1.1, 1.1)$	0.018	0	0	0	0	0	0

Table 5: Inferred posterior probabilities $P(Q - Q_0)$ in Zachary's karate club network. Results of runs made with RJMCMCWL. a is the Dirichlet hyper-parameter of the component weights, $\boldsymbol{\alpha}$; b is the Beta hyper-parameter of the probabilities of edges between components, $\boldsymbol{\pi}$, and is also the parameter of the Beta distribution from which the probabilities of edges with a vertex in the group created by a birth move are sampled; c and d are the parameters of the Beta distributions from which random variables are drawn for the reallocation of $\boldsymbol{\alpha}$ in the splitting and birth moves respectively.

66-Neuron Module	82-Neuron Module	24-Neuron Module	18-Neuron Module
F25B5.2 (42)	F25B5.2 (69)	unc-17 (24)	unc-17 (16)
flt-1 (56)	flt-1 (80)	cha-1 (24)	cha-1 (16)
cam-1 (35)		unc-32 (21)	unc-32 (16)
		C33A11.4 (21)	C33A11.4 (16)
		unc-58 (21)	unc-58 (16)
		flt-1 (21)	flt-1 (18)
		unc-40 (21)	unc-40 (16)
		unc-73 (21)	unc-73 (16)
		egl-21 (21)	egl-21 (16)
		mig-1 (21)	mig-1 (16)
		ace-1 (21)	ace-1 (16)
		mdl-1 (21)	mdl-1 (16)
		unc-1 (21)	unc-1 (16)
		unc-18 (21)	unc-18 (16)
		nlp-21 (21)	nlp-21 (16)
		unc-3 (21)	unc-3 (16)
		unc-8 (14)	unc-8 (16)
		unc-2 (13)	unc-2 (16)
		tba-1 (13)	tba-1 (16)
		trp-1 (13)	trp-1 (16)
		unc-5 (13)	dbl-1 (17)
		mec-12 (13)	tba-2 (16)
		unc-53 (13)	acr-5 (16)
		unc-4 (16)	ceh-12 (10)
		syg-1 (13)	

Table 6: Genes expressed in the neurons composing the 4 largest modules. The numbers following the gene name indicates the number of neurons where the gene is expressed in each module. Only genes expressed in more than half the neurons in each component are listed.