PROCEEDINGS A

rspa.royalsocietypublishing.org

Research

Check for updates

Article submitted to journal

Keywords:
Networks, message passing, belief propagation, percolation

Author for correspondence:
Mark Newman
e-mail: mejn@umich.edu

Message passing methods on complex networks

M. E. J. Newman

Department of Physics and Center for the Study of Complex Systems, University of Michigan, Ann Arbor, MI 48109. U.S.A.

Networks and network computations have become a primary mathematical tool for analysing the structure of many kinds of complex systems, ranging from the Internet and transportation networks to biochemical interactions and social networks. A common task in network analysis is the calculation of quantities that reside on the nodes of a network, such as centrality measures, probabilities or model states. In this perspective article we discuss message passing methods, a family of techniques for performing such calculations, based on the propagation of information between the nodes of a network. We introduce the message passing approach with a series of examples, give some illustrative applications and results, and discuss the deep connections between message passing and phase transitions in networks. We also point out some limitations of the message passing approach and describe some recently-introduced methods that address these limitations.

THE ROYAL SOCIETY PUBLISHING

1. Introduction

A network, for the purposes of this paper, is a set of points joined together in pairs by lines, as shown in Fig. 1. In the nomenclature of the field the points are called *nodes* or *vertices* and the lines are called *edges*. Networks provide a convenient tool for representing the pattern of connections or interactions within complex systems of many types. The Internet, for example, is a network of computers joined by data communications. The Web is a network of pages connected by hyperlinks. Biochemical networks, such as metabolic networks and protein interaction networks, are networks of chemicals and their interactions. And there is a long history of the quantitative study of social networks, which means not only modern online networks like Facebook and Twitter, but also offline social networks such as networks of friendship, collaboration and acquaintance, and the contact networks over which diseases spread. Figure 1, for instance, shows the patterns of collaboration among a group of scientists. In recent decades a large body of knowledge has built up concerning methods for analysing and interpreting network data, and databases have been assembled containing data sets and measurements for thousands of different networks. For an introduction to this vibrant and fascinating field see Refs. [1–5].

Development of the theory and practice of network analysis has been a multidisciplinary undertaking spanning mathematics, computer science, physics, statistics, the social sciences and other fields and has incorporated techniques from many areas, including linear algebra and spectral theory, probability and combinatorics, randomized models, statistics and machine learning, random matrix theory and a wide range of numerical methods. In this paper we focus on a less well known but powerful class of methods variously called *message passing*, *belief propagation* or *cavity methods*. These methods have theoretical foundations going back to the 1930s [6] but in their modern incarnation they were first formulated by Pearl in the 1980s [7]. They have been widely applied to problems in computer science and, less commonly, statistical physics and other fields [8–10]. In the network applications we consider here the first goal of message passing methods is to calculate properties of individual nodes in networks, such as the state they are in at a given time, but message passing can also be applied to the calculation of global network properties and can serve as a starting point for further analyses, for example of structural phase transitions in networks.

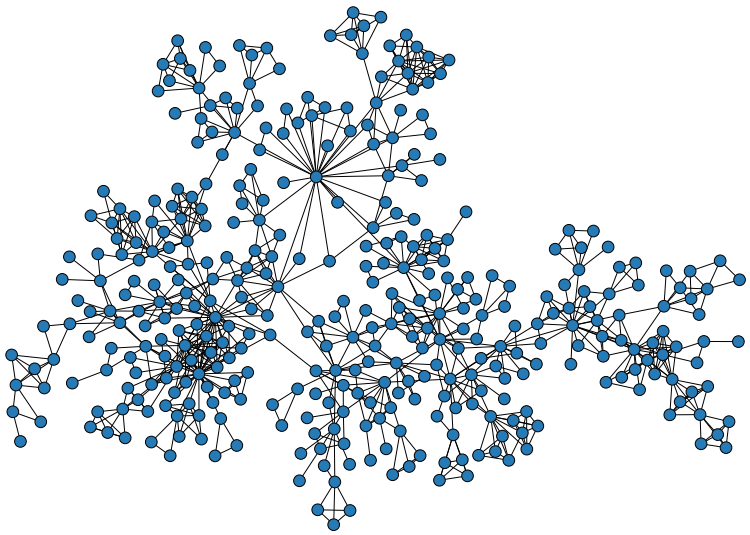

Figure 1. A small social network representing the pattern of collaborations among a group of 379 scientists.

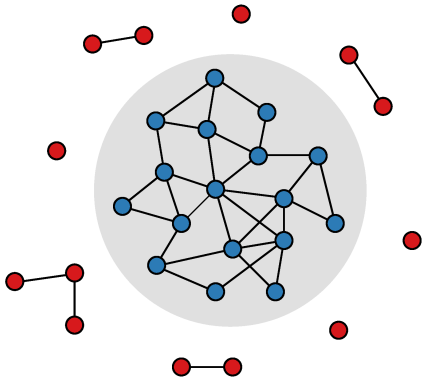

Figure 2. Most networks consist of one large group or giant component of connected nodes (circled) and multiple small components or unconnected single nodes.

We begin in the following section by introducing message passing on networks with some simple examples, then in Section 3 we show how the message passing equations for a network define a dynamical system whose fixed points and bifurcations correspond to phase transitions in the network. In Section 4 we give a number of further examples of the method, illustrating its use both as a computational tool and as a doorway to understanding phase transitions. In Section 5 we discuss a fundamental limitation of traditional message passing methods—that they are exact only on locally loop-free networks—and we describe a recently introduced approach that sidesteps this limitation, allowing message passing to be applied to a wide variety of networks, no matter what their structure. In Section 6 we offer our conclusions.

2. Message passing

We begin our discussion with a simple example to demonstrate the message passing method. Figure 2 illustrates a prominent feature of almost all networks, the so-called *giant component*. In most networks a significant fraction of the nodes are connected to form a large contiguous cluster or giant component, as shown in the centre of the figure, while the remainder of the nodes are grouped into a number of small components, shown around the edges. Networks do not normally contain more than one giant component because the chances of two such components existing yet not being connected together is vanishingly slim. At the same time a network with no giant component is, for most practical purposes, not really a network at all and so would not be studied in the first place. As a result, almost every network of scientific or technological importance has the qualitative structure shown in Fig. 2.

Let us consider the following question: given a complete network of n nodes and a specified node within that network, does the node in question belong to the giant component? To be more precise, we will define the giant component to be the largest component of connected nodes in the network (a slight abuse of nomenclature, but one that will cause us no problems here). There exist simple computer algorithms that will traverse a network and construct all of its components, allowing us to answer our question quickly, but here, for illustrative purposes, we will take a different approach.

Let the n nodes in our network be labelled by $i = 1 \ldots n$ in any convenient order and let μ_i be the probability that node i does *not* belong to the giant component. This is a somewhat trivial example, since μ_i is always either 0 or 1—either the node belongs to the giant component or it doesn't—but it is a useful example nonetheless and we will soon get to more complex ones.

The crucial point to notice is that node i is not in the giant component if and only if none of its network neighbours are in the giant component. If even one of its neighbours belongs to the giant

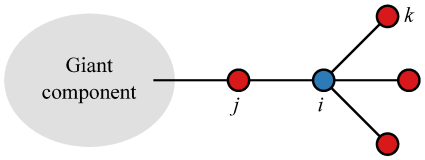

Figure 3. Node i is connected to the giant component via its neighbour j. Its other neighbour k is also connected to the giant component, but only via i, so k cannot be responsible for i's connection to the giant component.

component then i also belongs. Hence we might write

$$\mu_i = \prod_{j \in \mathcal{N}_i} \mu_j, \tag{2.1}$$

where $\mathcal{N}_i$ is the set of nodes that are neighbours of i. This equation says the probability that i is not in the giant component is equal to the probability that none of its neighbours are.

Trivial though it seems, however, this equation is wrong for two reasons, one obvious and one more subtle. The first and more obvious reason is that it assumes independence of the neighbours. In multiplying the probabilities μ_j we are assuming that the membership of one neighbour in the giant component is independent of the membership of another. This is violated if, for instance, two neighbours are connected by an edge, in which case either both are in the giant component or neither is, so their probabilities μ_j are correlated. Normally when applying message passing methods one assumes independence, which may be correct, at least approximately, in some cases but definitely is not in others. We will, for the moment, make the same assumption of independence here, but in Section 5 we will show how to relax it and apply message passing to networks where the states of nodes are correlated.

Even assuming that probabilities are independent, however, there is another reason why Eq. (2.1) is incorrect. Consider Fig. 3, which shows a possible configuration of the network around node i. In this example, node i is connected to the giant component via one of its neighbours j and this means in turn that its other neighbour k is also connected to the giant component via i. Thus i has a neighbour k in the giant component but k is not, and cannot be, i's link to the giant component, since k is itself only connected via i. So the question of whether i is in the giant component is not merely a matter of whether i's neighbour is in the giant component: i's neighbour must be connected to the giant component by some route *other than via i itself*.

We can modify Eq. (2.1) to account for this by making one simple change: we remove node i from the network before applying the equation, which automatically disconnects any nodes whose only connection to the giant component was via i. Any remaining neighbours j who are still connected to the giant component must be connected by some other route and it is these neighbours that we care about.

We define a new quantity $\mu_{i \leftarrow j}$ to be the probability that node j is not in the giant component of the network when i is removed. Then the correct equation is

$$\mu_i = \prod_{j \in \mathcal{N}_i} \mu_{i \leftarrow j}. \tag{2.2}$$

This equation will now correctly tell us if i is in the giant component.

To use this equation we still need to compute $\mu_{i \leftarrow j}$ itself. The probability that j is not in the giant component after i has been removed from the network is equal to the probability that none of j's neighbours other than i is in the giant component. Hence

$$\mu_{i \leftarrow j} = \prod_{\substack{k \in \mathcal{N}_j \\ k \neq i}} \mu_{j \leftarrow k}. \tag{2.3}$$

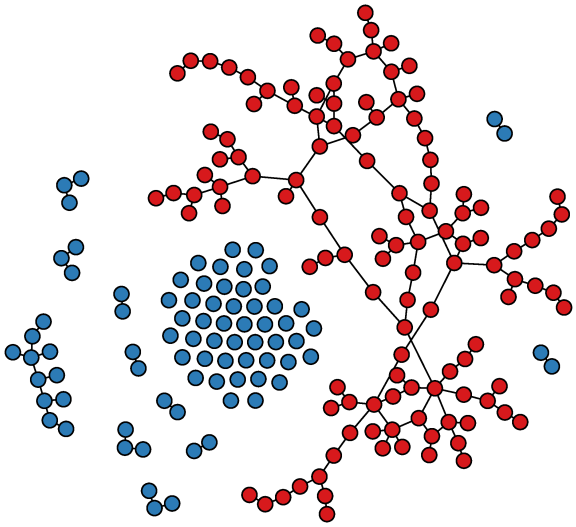

Figure 4. Application of the method of Eqs. (2.2) and (2.3) to a small network. The converged values of μ_i are all either zero or one and the colours of the nodes indicate the values.

This is a *message passing equation*. We can think of the probability $\mu_{i\leftarrow j}$ as a "message" that node i receives from its neighbour j. Node j transmits the probability that it is not in the giant component. This probability is in turn calculated from the messages that j receives from its other neighbours according to Eq. (2.3).

There are two equations of the form (2.3) for every edge in the network—one going in either direction along the edge—for a total of $2m$ equations, where m is the number of edges in the network. If we can solve these $2m$ equations for the $2m$ variables $\mu_{i\leftarrow j}$ then we can substitute the results into Eq. (2.2) and calculate the probabilities μ_i. In practice, the solution is normally computed by simple iteration. We choose any suitable set of starting values for the $\mu_{i\leftarrow j}$, such as random values in $(0, 1)$ for instance, and iterate (2.3) to convergence.

Figure 4 shows what happens when we do this on a small example network. The colours of the nodes in the figure represent the values of μ_i for each node and, as expected, there are only two values in this case, zero and one, indicating whether a node is in the giant component or not. As we can see, the calculation has correctly identified all the nodes in the largest component of the network.

(a) A more substantial example: Percolation

The example of the previous section illustrates the basic message passing approach, but it is relatively trivial. Let us see how to apply the method to a less trivial example. *Percolation* is a random process used, among other things, to model of the spread of disease over contact networks and the robustness of networks to failure of their connections [5,11,12]. In the percolation process we "occupy" edges in a network uniformly at random with some probability p and observe the clusters of nodes connected together by the occupied edges. (Technically this is "edge percolation." One can also study node percolation, where it is the nodes that are occupied, but we will not do so here.) If p is small there will be only small clusters, but for large enough p there will be one (and only one) giant cluster plus, potentially, some small clusters as well. In between these two regimes lies the percolation transition, the value p_c at which a giant cluster first forms. Because of the random nature of the edge occupation process, there can be some fluctuation in p_c, but the value becomes more and more narrowly concentrated as the size of the network grows.

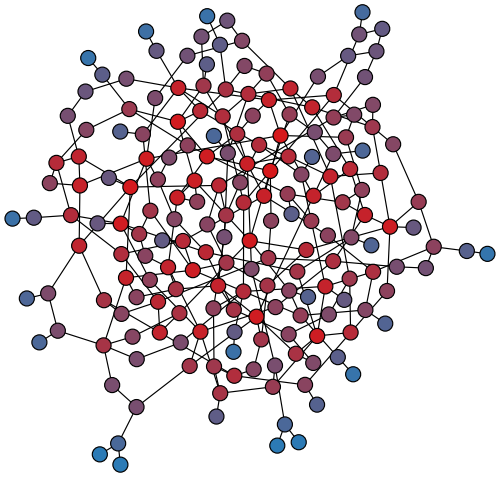

Figure 5. Application of Eqs. (2.4) and (2.5) to a small network. The colours of the nodes indicate the probability of belonging to the giant percolation cluster, calculated from μ_i, with red corresponding to higher probability and blue to lower.

It is only a small step from the methods of the previous section to the calculation of the probability that a node in a network belongs to the giant percolation cluster for given p. And because percolation is a probabilistic process, this is now a true probability, taking not just the values zero and one but any value in between. Our presentation follows the line of argument given by Karrer *et al.* [13] and Hamilton and Pryadko [14].

Let $\mu_{i\leftarrow j}$ be the probability that node j does not belong to the giant cluster if i is removed from the network. There are two ways for i to not be connected to the giant cluster via its neighbour j. The first is that the edge between i and j is unoccupied, which happens with probability $1-p$. The second is that the edge is occupied (probability p) but j is itself not in the giant cluster (probability $\mu_{i\leftarrow j}$). So the total probability is $1-p+p\mu_{i\leftarrow j}$ and the overall probability μ_i that i is not in the giant cluster is given by a product over neighbours of i thus:

$$\mu_i = \prod_{j\in\mathcal{N}_i} (1-p+p\mu_{i\leftarrow j}). \tag{2.4}$$

Now we repeat the same argument to calculate $\mu_{i\leftarrow j}$ itself. The probability that j is not in the giant cluster when i is removed from the network is equal to the probability that j is not connected to the giant cluster via any of its neighbours other than i:

$$\mu_{i\leftarrow j} = \prod_{\substack{k\in\mathcal{N}_j\\ k\neq i}} (1-p+p\mu_{j\leftarrow k}). \tag{2.5}$$

These are the message passing equations for percolation on a network and again we solve them by iteration from any suitable initial condition. Figure 5 shows the result for a small network. The colours of the nodes in this figure indicate the probability that a node belongs to the giant cluster. Note how the nodes around the periphery of the network have lower probability than those in the centre.

Unlike the giant component calculation of Section 2, this percolation calculation offers something that cannot be achieved with simpler numerical means. There exist algorithms that can find the giant cluster in a percolation system, but they do so only for one individual realization of the randomly occupied edges. If one wanted to calculate the probability μ_i of being in the giant cluster, one would have to run such an algorithm many times for many random realizations and average the results, which would be a complicated and computationally demanding process, and

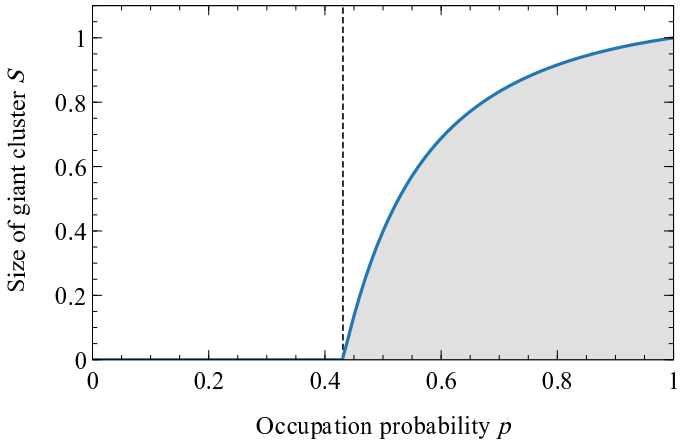

Figure 6. The size of the giant cluster as a function of occupation probability p for edge percolation on the network of Fig. 5, calculated using the message passing method of Eqs. (2.4), (2.5) and (2.6). The vertical dashed line indicates the expected position of the percolation transition from Eq. (3.4).

one moreover that would yield only an approximate answer, subject to statistical sampling error. Message passing on the other hand directly yields the value of μ_i, averaged over all realizations of the randomness, in a single calculation.

One can also extend the method to calculate other properties of the percolation process, including global properties. For instance, we can calculate the average size of the giant cluster as follows. We define a variable s_i that is 1 if i belongs to the giant cluster and 0 if it does not. Then the probability distribution of this variable is $(1-\mu_i)^{s_i}\mu_i^{1-s_i}$ and the size of giant cluster as a fraction of network size is $(1/n)\sum_i s_i$. Hence the expected size S of the giant cluster is

$$S = \sum_{\{s_i\}}\left[\frac{1}{n}\sum_i s_i\right]\left[\prod_i (1-\mu_i)^{s_i}\mu_i^{1-s_i}\right] = \frac{1}{n}\sum_i \sum_{s_i=0,1} s_i (1-\mu_i)^{s_i}\mu_i^{1-s_i}$$
$$= 1 - \frac{1}{n}\sum_{i=1}^{n}\mu_i. \tag{2.6}$$

Figure 6 shows the value of this quantity as a function of p for the same small network as in Fig. 5. The figure displays the classic percolation transition behaviour, with no giant cluster for small p and an abrupt phase transition—around $p = 0.43$ in this case—at which a giant cluster appears. We study this phase transition in detail in the following section.

3. Phase transitions

The message passing method is a useful tool for numerical calculation of node properties on networks as well as certain global quantities. As mentioned in the introduction, however, message passing can also be used as the foundation for further analytic calculations, especially with regard to phase transitions in networks. In this section we illustrate this point for the case of percolation.

As can be seen in Fig. 6, percolation shows a phase transition at a critical occupation probability p_c between a percolating state in which there exists a giant cluster and a non-percolating state with no giant cluster and small clusters only. When we are below this transition, where $p < p_c$, the probability $\mu_{i\leftarrow j}$ that node j is not in the giant cluster is, by definition, 1 for all i, j. Looking at the message passing equations, Eq. (2.5), we see that this is in fact a solution of the equations for any value of p—setting $\mu_{i\leftarrow j} = 1$ for all i, j just gives "$1 = 1$". So does this

mean that there is never a giant cluster in the network? It does not, because the outcome of the message passing calculation depends not only on whether this solution exists but also on whether the calculation actually returns this solution, versus some other solution.

We can think of the iteration of Eq. (2.5) as a discrete-time dynamical system that moves us around the space defined by the set of probabilities $\mu_{i\leftarrow j}$. The crucial question we need to answer is whether this process converges to the trivial solution $\mu_{i\leftarrow j} = 1$. If it does, then there is no giant cluster in the network. If it converges to some other (nontrivial) solution, then there is a giant cluster. Another way to say the same thing is that there is no giant cluster if the trivial solution is a stable fixed point of the iteration. If it is unstable then there is a giant cluster.

This now gives us a simple method for determining whether there is a giant cluster: we linearize to find the stability of the trivial fixed point. We write $\mu_{i\leftarrow j} = 1 - \epsilon_{i\leftarrow j}$ (with a negative sign since $\mu_{i\leftarrow j} \le 1$) and substitute into Eq. (2.5) to get

$$1 - \epsilon_{i\leftarrow j} = \prod_{\substack{k\in\mathcal{N}_j \\ k\neq i}} (1 - p\epsilon_{j\leftarrow k}) = 1 - p \sum_{\substack{k\in\mathcal{N}_j \\ k\neq i}} \epsilon_{j\leftarrow k} + \mathrm{O}(\epsilon^2). \tag{3.1}$$

Hence, close to the fixed point we have the linear form

$$\epsilon_{i\leftarrow j} = p \sum_{\substack{k\in\mathcal{N}_j \\ k\neq i}} \epsilon_{j\leftarrow k}. \tag{3.2}$$

Alternatively we can write this in matrix format as

$$\boldsymbol{\epsilon} = p\mathbf{B}\boldsymbol{\epsilon}, \tag{3.3}$$

where $\boldsymbol{\epsilon}$ is the $2m$-element vector with elements $\epsilon_{i\leftarrow j}$ and $\mathbf{B}$ is a $2m \times 2m$ non-symmetric matrix with one row and column for each directed edge $i \leftarrow j$ and elements chosen so as to correctly reproduce Eq. (3.2). By inspection, this requires $B_{i\leftarrow j,k\leftarrow l} = \delta_{jk}(1 - \delta_{il})$. The resulting matrix, known as the Hashimoto edge-incidence matrix or more commonly the *non-backtracking matrix*, appears in a number of contexts in network theory, including the study of community detection algorithms [15] and centrality metrics [16].

The trivial fixed point is unstable if a small $\boldsymbol{\epsilon}$ grows under the iteration of Eq. (3.3), which is equivalent to saying that $p\lambda > 1$, where λ is the leading eigenvalue of $\mathbf{B}$. Thus there will be a giant cluster if and only if $p > 1/\lambda$ and hence we conclude that

$$p_c = \frac{1}{\lambda} \tag{3.4}$$

is the percolation threshold on the network [13,14]. Applying this approach to the network of Fig. 5 the resulting value of p_c is shown as the vertical dashed line in Fig. 6 and closely matches the apparent phase transition point at which the giant cluster appears.

Thus by considering the properties of the message passing equations as a dynamical system we have been able to derive a new formal result about the position of the percolation phase transition on networks.

4. Other applications of message passing

Message passing can be applied to a wide range of other calculations on networks. We give three illustrative examples in this section.

(a) The Ising model

The large class of finite-temperature spin models in physics, which includes the Ising model, the Heisenberg model, the XY model, and spin-glass and random-field models, can be treated using message passing methods [8,17]. Let us take as an example the Ising model in zero magnetic field,

which is defined on a general network by the Hamiltonian

$$H = -\tfrac{1}{2}\sum_{ij} A_{ij} s_i s_j, \tag{4.1}$$

where $s_i = \pm 1$ is an Ising spin on node i and A_{ij} is an element of the adjacency matrix $\mathbf{A}$ of the network with value $A_{ij} = 1$ if there exists an edge between nodes i and j and 0 otherwise.

We can usefully rewrite this Hamiltonian by "centring" it around a particular node i thus:

$$\begin{aligned} H &= -\sum_{j\in\mathcal{N}_i}\Big[s_i s_j + \sum_{\substack{k\in\mathcal{N}_j\\ k\neq i}}\Big[s_j s_k + \sum_{\substack{l\in\mathcal{N}_k\\ l\neq j}}\Big[s_k s_l + \ldots\Big]\Big]\Big] \\ &= -\sum_{j\in\mathcal{N}_i}\big[s_i s_j + h_{i\leftarrow j}(s_j)\big], \end{aligned} \tag{4.2}$$

where

$$h_{i\leftarrow j}(s_j) = \sum_{\substack{k\in\mathcal{N}_j\\ k\neq i}}\Big[s_j s_k + \sum_{\substack{l\in\mathcal{N}_k\\ l\neq j}}\Big[s_k s_l + \ldots\Big]\Big] = \sum_{\substack{k\in\mathcal{N}_j\\ k\neq i}}\big[s_j s_k + h_{j\leftarrow k}(s_k)\big]. \tag{4.3}$$

Equation (4.2) correctly counts each interaction $s_i s_j$ exactly once, so long as no nodes appear in more than one of the sums, which is equivalent to saying that there are no paths between the spins around the central node i, other than via i itself. As with our percolation example, we will, for the moment, assume this to be the case. In Section 5 we show how to remove this assumption. Note also that while our notation $h_{i\leftarrow j}(s_j)$ explicitly includes only the dependence on s_j, the value of $h_{i\leftarrow j}(s_j)$ does depend on other spins as well. The latter however will be summed over shortly and so will play no further role.

As we have said, message passing methods are particularly useful for calculating properties of individual nodes. Let us take as an example the calculation of the probability within the Ising model that the spin s_i has a particular value $r = \pm 1$. At inverse temperature β this probability is given by the Boltzmann average

$$P[s_i = r] = \frac{1}{Z}\sum_{\{s_i\}} \mathbb{1}_{s_i = r}\, \mathrm{e}^{-\beta H}, \tag{4.4}$$

where $Z = \sum_{\{s_i\}} \mathrm{e}^{-\beta H}$ is the partition function and $\mathbb{1}_x$ is the indicator function which is 1 if x is true and 0 otherwise. Using Eq. (4.2), the sum in the numerator can be expanded as

$$\begin{aligned} \sum_{\{s_i\}} \mathbb{1}_{s_i = r}\, \mathrm{e}^{-\beta H} &= \sum_{\{s_i\}} \exp\Big(\beta \sum_{j\in\mathcal{N}_i}\big[r s_j + h_{i\leftarrow j}(s_j)\big]\Big) = \prod_{j\in\mathcal{N}_i}\prod_{\substack{k\notin\mathcal{N}_i\\ k\neq i}}\sum_{s_j}\sum_{s_k} \mathrm{e}^{\beta[r s_j + h_{i\leftarrow j}(s_j)]} \\ &= \prod_{j\in\mathcal{N}_i}\sum_{s_j} \mathrm{e}^{\beta r s_j}\prod_{\substack{k\notin\mathcal{N}_i\\ k\neq i}}\sum_{s_k} \mathrm{e}^{\beta h_{i\leftarrow j}(s_j)}. \end{aligned} \tag{4.5}$$

Now we define a message $\mu^s_{i\leftarrow j}$ according to

$$\mu^s_{i\leftarrow j} = \frac{1}{Z_{i\leftarrow j}}\prod_{\substack{k\notin\mathcal{N}_i\\ k\neq i}}\sum_{s_k} \mathrm{e}^{\beta h_{i\leftarrow j}(s)}, \tag{4.6}$$

where $Z_{i\leftarrow j}$ is a normalizing factor chosen to make $\sum_{s=\pm 1}\mu^s_{i\leftarrow j} = 1$. Then

$$P[s_i = r] = \frac{1}{Z}\prod_{j\in\mathcal{N}_i}\sum_{s} \mathrm{e}^{\beta r s} Z_{i\leftarrow j}\mu^s_{i\leftarrow j} = \frac{1}{Z_i}\prod_{j\in\mathcal{N}_i}\big(\mathrm{e}^{\beta r}\mu^+_{i\leftarrow j} + \mathrm{e}^{-\beta r}\mu^-_{i\leftarrow j}\big), \tag{4.7}$$

with $\mu^\pm_{i\leftarrow j}$ denoting the value of $\mu^s_{i\leftarrow j}$ for $s = \pm 1$ and

$$Z_i = \frac{Z}{\prod_{j\in\mathcal{N}_i} Z_{i\leftarrow j}}. \tag{4.8}$$

The partition function Z is difficult to calculate directly, but evaluating (4.7) only requires Z_i, which can be done easily by noting that $\sum_{r=\pm1} P[s_i = r] = 1$, so

$$Z_i = \sum_{r=\pm1} \prod_{j\in\mathcal{N}_i} \bigl(\mathrm{e}^{\beta r}\mu^+_{i\leftarrow j} + \mathrm{e}^{-\beta r}\mu^-_{i\leftarrow j}\bigr). \tag{4.9}$$

To use Eq. (4.7) we still need the values of the messages $\mu^s_{i\leftarrow j}$, which we can calculate from Eqs. (4.3) and (4.6) thus:

$$\begin{aligned}\mu^r_{i\leftarrow j} &= \frac{1}{Z_{i\leftarrow j}} \prod_{\substack{k\notin\mathcal{N}_i\\ k\neq i}} \sum_{s_k} \mathrm{e}^{\beta h_{i\leftarrow j}(r)} = \frac{1}{Z_{i\leftarrow j}} \prod_{\substack{k\in\mathcal{N}_j\\ k\neq i}} \prod_{\substack{l\notin\mathcal{N}_j\\ l\neq j}} \sum_{s_k}\sum_{s_l} \mathrm{e}^{\beta[rs_k + h_{j\leftarrow k}(s_k)]}\\ &= \frac{1}{Z_{i\leftarrow j}} \prod_{\substack{k\in\mathcal{N}_j\\ k\neq i}} \sum_{s_k} \mathrm{e}^{\beta r s_k} \prod_{\substack{l\notin\mathcal{N}_j\\ l\neq j}} \sum_{s_l} \mathrm{e}^{\beta h_{j\leftarrow k}(s_k)}\\ &= \frac{1}{Z_{i\leftarrow j}} \prod_{\substack{k\in\mathcal{N}_j\\ k\neq i}} \sum_{s} \mathrm{e}^{\beta r s} \mu^s_{j\leftarrow k},\end{aligned} \tag{4.10}$$

or

$$\mu^r_{i\leftarrow j} = \frac{1}{Z_{i\leftarrow j}} \prod_{\substack{k\in\mathcal{N}_j\\ k\neq i}} \bigl(\mathrm{e}^{\beta r}\mu^+_{j\leftarrow k} + \mathrm{e}^{-\beta r}\mu^-_{j\leftarrow k}\bigr). \tag{4.11}$$

Since $Z_{i\leftarrow j}$ is defined to make $\sum_{r=\pm1}\mu^r_{i\leftarrow j} = 1$, its value is given by

$$Z_{i\leftarrow j} = \sum_{r=\pm1} \prod_{\substack{k\in\mathcal{N}_j\\ k\neq i}} \bigl(\mathrm{e}^{\beta r}\mu^+_{j\leftarrow k} + \mathrm{e}^{-\beta r}\mu^-_{j\leftarrow k}\bigr). \tag{4.12}$$

Equations (4.11) and (4.12) define the message passing process for the Ising model and as before can be solved by simple iteration, starting for instance from random values. Once the values converge, we can calculate the probability that $s_i = \pm1$ from Eq. (4.7), and from these probabilities we can calculate other quantities of interest. For example, the average magnetization $m_i = \langle s_i\rangle$ at node i is given by $m_i = P[s_i = +1] - P[s_i = -1]$ and from this one can calculate the global average magnetization $m = (1/n)\sum_i m_i$.

The message passing equations also give us a way to calculate the partition function Z of the Ising model, something that is difficult to do by other methods such as Monte Carlo simulation. Rearranging Eq. (4.8) we have

$$Z = Z_i \prod_{j\in\mathcal{N}_i} Z_{i\leftarrow j}, \tag{4.13}$$

so we can calculate the partition function directly from the quantities Z_i and $Z_{i\leftarrow j}$, which are evaluated as part of the message passing process. From Z we can then calculate other quantities such as the free energy, which is $F = -\beta^{-1}\log Z = -\beta^{-1}(\log Z_i + \sum_{j\in\mathcal{N}_i}\log Z_{i\leftarrow j})$ or, if we prefer a more symmetric formulation,

$$F = -\frac{1}{n\beta}\biggl(\sum_i \log Z_i + \sum_{i\leftarrow j} \log Z_{i\leftarrow j}\biggr), \tag{4.14}$$

where the second sum is over all directed edges $i \leftarrow j$ in the network.

Figure 7 shows results for average magnetization calculated by message passing on a small network. The Ising model in zero field is up-down symmetric, so one might imagine the value of m would just be zero, but this is not the case. The message passing calculation displays spontaneous symmetry breaking, just as the Ising model itself does. Zero magnetization implies that all spins are equally likely to be up or down, so $\mu^+_{i\leftarrow j} = \mu^-_{i\leftarrow j} = \frac{1}{2}$ for all i, j, and Eq. (4.11) does have such a solution, but it also has two other solutions, one where we have $\mu^+_{i\leftarrow j} > \frac{1}{2}$ and $\mu^-_{i\leftarrow j} < \frac{1}{2}$ on average and one where we have the reverse, and for these solutions m is nonzero. As

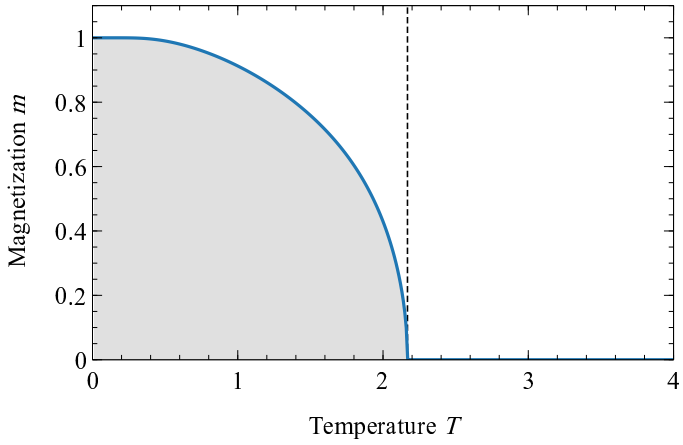

Figure 7. The magnetization of an Ising model on a small network as a function of temperature $T = 1/\beta$, calculated from Eqs. (4.7) and (4.11). The vertical dashed line indicates the expected position of the ferromagnetic phase transition, calculated from the leading eigenvalue of the non-backtracking matrix using Eq. (4.17).

a simple example, in the low-temperature limit $\beta \to \infty$ it is straightforward to see that $\mu^{\pm}_{i\leftarrow j} = 1$ and 0 are solutions, which correspond to all spins up and all spins down.

Which solution the message passing process converges to depends on whether the solutions are stable or unstable under the iteration of Eq. (4.11). In particular, if the symmetric solution at $\mu^{+}_{i\leftarrow j} = \mu^{-}_{i\leftarrow j} = \frac{1}{2}$ is unstable then we must converge to one of the other, symmetry-broken solutions and hence we get a nonzero magnetization, indicating that we are below the symmetry-breaking phase transition, in the ferromagnetic regime of the model. Conversely, if we converge to the symmetric solution then we are above the transition. As with percolation, therefore, the point at which the symmetric solution for the Ising model undergoes a bifurcation from stable to unstable corresponds to the phase transition.

We can check for stability by once again expanding about the symmetric point. Observing that $\mu^{-}_{i\leftarrow j} = 1 - \mu^{+}_{i\leftarrow j}$ and putting $\mu^{\pm}_{i\leftarrow j} = \frac{1}{2}(1 \pm \epsilon_{i\leftarrow j})$ in Eq. (4.11) we have

$$\begin{aligned}
\tfrac{1}{2}(1+\epsilon_{i\leftarrow j}) &= \frac{1}{Z_{i\leftarrow j}} \prod_{\substack{k\in\mathcal{N}_j \\ k\neq i}} \tfrac{1}{2}\left[\mathrm{e}^{\beta}(1+\epsilon_{j\leftarrow k}) + \mathrm{e}^{-\beta}(1-\epsilon_{j\leftarrow k})\right] \\
&= \frac{1}{Z_{i\leftarrow j}} \prod_{\substack{k\in\mathcal{N}_j \\ k\neq i}} \left(\cosh\beta + \epsilon_{j\leftarrow k}\sinh\beta\right) \\
&= \frac{(\cosh\beta)^{|\mathcal{N}_j|-1}}{Z_{i\leftarrow j}} \left[1 + \tanh\beta \sum_{\substack{k\in\mathcal{N}_j \\ k\neq i}} \epsilon_{j\leftarrow k}\right] + \mathrm{O}(\epsilon^2),
\end{aligned} \tag{4.15}$$

where $|\mathcal{N}_j|$ is the degree of node j, i.e., the number of its neighbours. Making the same substitution in Eq. (4.12) gives just $Z_{i\leftarrow j} = 2(\cosh\beta)^{|\mathcal{N}_j|-1}$, and hence, neglecting terms of order ϵ^2, the linearized message passing equations are

$$\epsilon_{i\leftarrow j} = \tanh\beta \sum_{\substack{k\in\mathcal{N}_j \\ k\neq i}} \epsilon_{j\leftarrow k}. \tag{4.16}$$

This has the same form as Eq. (3.2) for the percolation case and carries the same implication: just as the critical occupation probability for percolation satisfies $\lambda p_c = 1$, where λ is the leading

eigenvalue of the non-backtracking matrix, so the critical temperature of the Ising model satisfies $\lambda \tanh \beta_c = 1$, or

$$\beta_c = \operatorname{arctanh} \frac{1}{\lambda}. \tag{4.17}$$

The corresponding value of the critical temperature $T_c = 1/\beta_c$ is marked as the vertical dashed line in Fig. 7 and agrees nicely with the apparent transition point between $m > 0$ and $m = 0$.

(b) Graph spectra

The eigenvalue spectrum of the adjacency matrix of a graph or network plays a central role in several aspects of the theory of networks, including the calculation of centrality measures [18], behaviour of dynamical systems on networks [19], percolation properties [13], community detection [20] and network epidemiology [5]. Complete matrix spectra can be calculated numerically by standard means such as the QR algorithm [21], but this approach is slow and limited to relatively small networks, up to about 10 000 nodes. As another example of message passing, we here show how one can use the method to calculate graph spectra in very competitive running times, especially on sparse graphs. Our presentation follows that of [22,23].

One can write the spectrum of a network in terms of the spectral density

$$\rho(x) = \frac{1}{n} \sum_{i=1}^{n} \delta(x - \lambda_i), \tag{4.18}$$

where λ_i are the eigenvalues of the adjacency matrix and $\delta(x)$ is the Dirac delta function. There are a variety of analytic forms for the delta function, but for our purposes a useful one is the Lorentzian or Cauchy distribution in the limit of zero width:

$$\delta(x) = \lim_{\eta \to 0^+} \frac{\eta/\pi}{x^2 + \eta^2} = -\frac{1}{\pi} \lim_{\eta \to 0^+} \operatorname{Im} \frac{1}{x + \mathrm{i}\eta}, \tag{4.19}$$

where $\eta \to 0^+$ means that η tends to zero from above. Substituting for $\delta(x)$ in Eq. (4.18) we then have

$$\rho(x) = -\frac{1}{n\pi} \lim_{\eta \to 0^+} \operatorname{Im} \sum_{i=1}^{n} \frac{1}{x - \lambda_i + \mathrm{i}\eta}. \tag{4.20}$$

In our calculations we will focus on the complex generalization of the spectral density

$$\rho(z) = -\frac{1}{n\pi} \sum_{i=1}^{n} \frac{1}{z - \lambda_i} = -\frac{1}{n\pi} \operatorname{Tr}(z\mathbf{I} - \mathbf{A})^{-1}, \tag{4.21}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix and $\mathbf{A}$ is the adjacency matrix as previously. If we can calculate this quantity then the real spectral density $\rho(x)$ of Eq. (4.20) is given by setting $z = x + \mathrm{i}\eta$, taking the imaginary part, and then taking the limit as $\eta \to 0^+$.

Expanding the matrix inverse in (4.21) as a power series in z and taking the trace term by term gives

$$\rho(z) = -\frac{1}{n\pi z} \sum_{r=0}^{\infty} \frac{\operatorname{Tr} \mathbf{A}^r}{z^r}. \tag{4.22}$$

But the trace of $\mathbf{A}^r$ is equal to the number of closed (potentially self-intersecting) walks of length r on a network—walks that start and end at the same node and traverse exactly r edges along the way. So if we can count closed walks we can evaluate Eq. (4.22), and hence calculate the spectral density of our network.

To count closed walks, we first consider the more limited problem of counting closed walks that start from a specified node and return to the same node for the first and only time on their final step. Walks that return only once like this we call *excursions*. Let us denote by $n_{i \leftarrow j}^{(r)}$ the number of excursions that start from node i, take their first step along the edge from i to j, and return to i after exactly r steps. If the neighbours of i are connected to one another other than via i

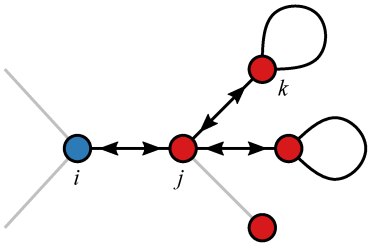

Figure 8. An excursion from node i traverses the edge from i to one of its neighbours j, then makes any number of further excursions via other neighbours k of node j before traversing the edge from j to i again and ending.

itself, either directly by edges or by other short paths, then it is possible for an excursion to return to i along a different edge from the one it left by. As with our other message passing calculations we will assume for the moment that this is not the case. (Again, we will show in Section 5 how to relax this assumption.) Thus, all excursions that start by traversing the edge (i, j) return along this same edge.

Generically, every excursion of length r now has the structure shown in Fig. 8: it first traverses the edge from i to j, then it makes some number m (possibly zero) of separate excursions from j via neighbours k (not including i) and then it traverses the edge from j to i back again and stops at i after a total of r steps. This observation allows us to express the number of excursions $n^{(r)}_{i\leftarrow j}$ self-consistently in the form

$$n^{(r)}_{i\leftarrow j} = \sum_{m=0}^{\infty} \Bigg[\sum_{\substack{k_1\in\mathcal{N}_j\\ k_1\neq i}} \cdots \sum_{\substack{k_m\in\mathcal{N}_j\\ k_m\neq i}}\Bigg] \Bigg[\sum_{r_1=1}^{\infty}\cdots\sum_{r_m=1}^{\infty}\Bigg] \mathbb{1}_{\sum_u r_u = r-2} \prod_{u=1}^{m} n^{(r_u)}_{j\leftarrow k_u}. \tag{4.23}$$

In this expression the product $\prod_{u=1}^{m} n^{(r_u)}_{j\leftarrow k_u}$ represents the number of combinations of excursions from j that have lengths $r_1 \ldots r_m$ and proceed via neighbours $k_1 \ldots k_m$ of j (not necessarily distinct). The lengths and first steps are summed over all possibilities and the indicator function ensures that only those combinations whose lengths add up to $r-2$ are allowed, the remaining two steps being reserved for traversing the edge (i, j) in either direction.

Now we define a message $\mu_{i\leftarrow j}(z)$ by

$$\begin{aligned}
\mu_{i\leftarrow j}(z) &= \sum_{r=1}^{\infty} \frac{n^{(r)}_{i\leftarrow j}}{z^r} \\
&= \sum_{r=1}^{\infty}\frac{1}{z^r}\sum_{m=0}^{\infty}\Bigg[\sum_{\substack{k_1\in\mathcal{N}_j\\ k_1\neq i}} \cdots \sum_{\substack{k_m\in\mathcal{N}_j\\ k_m\neq i}}\Bigg] \Bigg[\sum_{r_1=1}^{\infty}\cdots\sum_{r_m=1}^{\infty}\Bigg] \mathbb{1}_{\sum_u r_u = r-2} \prod_{u=1}^{m} n^{(r_u)}_{j\leftarrow k_u} \\
&= \frac{1}{z^2}\sum_{m=0}^{\infty}\Bigg[\sum_{\substack{k_1\in\mathcal{N}_j\\ k_1\neq i}} \cdots \sum_{\substack{k_m\in\mathcal{N}_j\\ k_m\neq i}}\Bigg] \Bigg[\sum_{r_1=1}^{\infty}\cdots\sum_{r_m=1}^{\infty}\Bigg] \prod_{u=1}^{m} \frac{n^{(r_u)}_{j\leftarrow k_u}}{z^{r_u}} \\
&= \frac{1}{z^2}\sum_{m=0}^{\infty}\prod_{u=1}^{m}\Bigg[\sum_{\substack{k_u\in\mathcal{N}_j\\ k_u\neq i}} \sum_{r_u=1}^{\infty} \frac{n^{(r_u)}_{j\leftarrow k_u}}{z^{r_u}}\Bigg] = \frac{1}{z^2}\sum_{m=0}^{\infty}\Bigg[\sum_{k\in\mathcal{N}_j, k\neq i}\mu_{j\leftarrow k}(z)\Bigg]^m \\
&= \frac{1/z^2}{1-\sum_{k\in\mathcal{N}_j,k\neq i}\mu_{j\leftarrow k}(z)}.
\end{aligned} \tag{4.24}$$

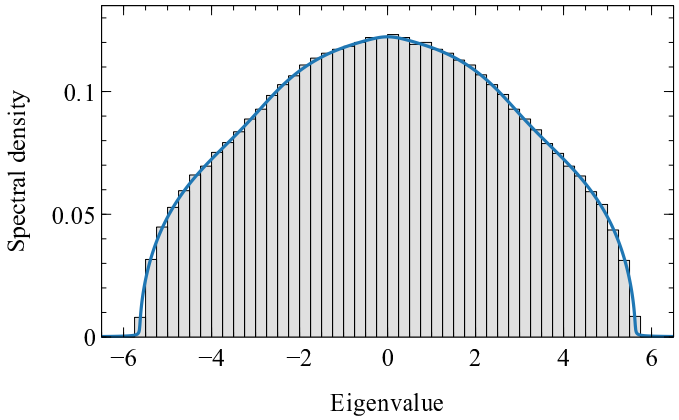

Figure 9. The spectral density of a 10 000-node network calculated in two ways: first using the standard QR algorithm to compute all eigenvalues and then calculating a histogram of the results (grey), and second using the message passing equations, (4.24) and (4.27), with $\eta = 0.01$ (solid curve).

This is our message passing equation, which we solve in the normal fashion by iteration. Once we have the values of the messages we can write the total number $n_i^{(r)}$ of closed walks of length r starting and ending at node i as a composition of any number m of excursions via the nodes in i's neighbourhood:

$$n_i^{(r)} = \sum_{m=0}^{\infty} \Biggl[\sum_{j_1\in\mathcal{N}_i} \cdots \sum_{j_m\in\mathcal{N}_i}\Biggr] \Biggl[\sum_{r_1=1}^{\infty} \cdots \sum_{r_m=1}^{\infty}\Biggr] \mathbb{1}_{\sum_u r_u = r} \prod_{u=1}^{m} n_{j\leftarrow k_u}^{(r_u)}. \tag{4.25}$$

Then, by a derivation similar to that of (4.24), we have

$$\sum_{r=0}^{\infty} \frac{\operatorname{Tr}\mathbf{A}^r}{z^r} = n + \sum_{r=1}^{\infty}\sum_{i=1}^{n} \frac{n_i^{(r)}}{z^r} = n + \sum_{i=1}^{n} \frac{1}{1-\sum_{j\in\mathcal{N}_i}\mu_{i\leftarrow j}(z)}. \tag{4.26}$$

And substituting this result into Eq. (4.22) and neglecting the initial $+n$, which will disappear anyway when we take the imaginary part, we get our expression for the complex spectral density

$$\rho(z) = -\frac{1}{n\pi z}\sum_{i=1}^{n} \frac{1}{1-\sum_{j\in\mathcal{N}_i}\mu_{i\leftarrow j}(z)}. \tag{4.27}$$

Taking the imaginary part now gives us $\rho(x)$.

This procedure gives us a complete recipe for calculating the spectral density of a network using message passing. We can use it for numerical calculations by setting $z = x + \mathrm{i}\eta$ for small but nonzero η, explicitly iterating to convergence the message passing equations of (4.24) with this value of z and taking the imaginary part of (4.27) to find the spectral density at x. Figure 9 shows an example application to a network of $n = 10\,000$ nodes. Also shown in the figure are the results of direct numerical calculations using the QR algorithm and the agreement between the two is good.

As with our other message passing methods, one can also use the equations for the spectral density as the basis for further analytic computations. To give a simple example, consider a random d-regular network, i.e., a network in which every node has d edges but the nodes are otherwise connected together at random. When the number of nodes n is large, the neighbourhood of every node in such a network looks identical: each node has d neighbours, each of which has d neighbours, and so on. This means that the messages $\mu_{i\leftarrow j}$ on all edges have

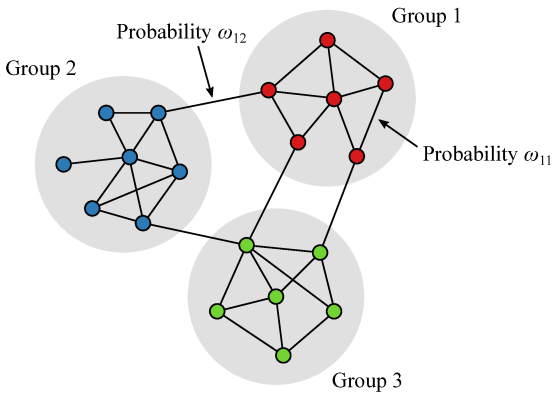

Figure 10. Many networks display community structure, in which the network divides into tightly-knit groups of nodes, with many connections within groups but few connections between them. The stochastic block model imitates this structure by placing edges randomly between node pairs with probabilities ω_{rs} that depend on the groups r, s that the nodes belong to.

the same value $\mu(z)$ for any z, so Eq. (4.27) becomes

$$\rho(z) = -\frac{1}{\pi z[1 - d\mu(z)]}\,, \tag{4.28}$$

and the message passing equations (4.24) take the simple form

$$\mu(z) = \frac{1/z^2}{1 - (d-1)\mu(z)}. \tag{4.29}$$

This last result can be rearranged into the quadratic equation $(d-1)\mu^2 - \mu + 1/z^2 = 0$ and, computing the solutions, substituting into Eq. (4.28) and taking the imaginary part, we find that

$$\rho(x) = (d/2\pi)\frac{\sqrt{4(d-1) - x^2}}{d^2 - x^2}. \tag{4.30}$$

This is the well-known spectrum of Kesten and McKay for the random regular graph [24], but derived here by a different—and shorter—method than the traditional one.

The same approach can be extended to other model networks as well. For instance, it can be easily modified for networks in which the number of edges at nodes—their degree—alternates between two different values, or more generally to the large class of "equitable random graphs," which can have nodes of many different degrees [25]. By making some further approximations, the same method can also be used to calculate the spectrum of the configuration model [23], perhaps the most important structural model in the theory of networks [26,27].

(c) Community detection

As our final example of message passing we look at an application to one of the classic problems in the study of networks: community detection. Many real-world networks have large-scale structure of the kind sketched in Fig. 10, in which the network nodes are divided into some number of groups or communities, with dense connections within groups but only sparser connections between groups. Structure of this kind often reflects functional divisions between nodes, and the ability to find such structure in unlabelled network data can be an invaluable tool for understanding the link between form and function in networked systems [20].

There are a large number of techniques for detecting communities in networks, but one of the most promising, and also most mathematically principled, makes use of a fit to a statistical model

of community structure, as illustrated in Fig. 10. Briefly, suppose we take n nodes and divide them among q groups numbered $1 \ldots q$. Then we place edges at random such that each pair of nodes is connected independently with probability ω_{rs} which depends on the groups r and s that the nodes belong to. Thus two nodes in group 1 would be connected with probability ω_{11} and nodes in groups 1 and 2 would be connected with probability ω_{12}. This model of a network is known as the *stochastic block model* [28], and if the diagonal probabilities ω_{rr} are larger than the off-diagonal ones it generates a random network with classic community structure of the kind shown in the figure.

The community detection method that we focus on here involves fitting this model (or one of its variants [29–31]) to observed network data by the method of maximum a posteriori probability. Suppose we observe a network with adjacency matrix $\mathbf{A} = [A_{ij}]$, which we hypothesize was generated from the stochastic block model. If s_i denotes the community to which node i belongs then the probability of generating this network is

$$P[\mathbf{A}|\mathbf{s}] = \prod_{i<j} \omega_{s_i s_j}^{A_{ij}} (1 - \omega_{s_i s_j})^{1-A_{ij}}, \tag{4.31}$$

where $\mathbf{s} = [s_i]$ is the vector of community assignments and the sum over values $i < j$ ensures that we count each node pair only once. By Bayes' rule we now have

$$P[\mathbf{s}|\mathbf{A}] = \frac{P[\mathbf{A}|\mathbf{s}]P[\mathbf{s}]}{P[\mathbf{A}]} = \frac{1}{Z} \prod_i \pi_{s_i} \prod_{i<j} \omega_{s_i s_j}^{A_{ij}} (1 - \omega_{s_i s_j})^{1-A_{ij}}, \tag{4.32}$$

where for convenience we have defined $Z = P[\mathbf{A}]$ and we assume a categorical prior $P[\mathbf{s}]$ on the group assignments in which each node is assigned to group s with prior probability π_s which we choose.

Fitting the model to the observed network involves finding the most likely values of both the community assignments s_i and the parameters π_r, ω_{rs} by maximizing (4.32). Finding the parameter values is a relatively straightforward task—it can be accomplished using a standard expectation-maximization (EM) algorithm. Finding the community assignments, on the other hand, is more difficult and it is this task that we address here, assuming that we already know the values of the parameters.

We focus on calculating the probability $q_i^s = P[s_i = r]$ that node i is assigned to group r. We will not go into the derivation of the message passing equations in detail—see Refs. [32,33] for a discussion—but, briefly, we define a set of messages $\mu_{i\leftarrow j}^r$ that satisfy the message passing equations

$$\mu_{i\leftarrow j}^r = \frac{\pi_r}{Z_{i\leftarrow j}} \prod_k \Bigl(1 - \sum_s q_k^s \omega_{rs}\Bigr) \prod_{\substack{k\in\mathcal{N}_j \\ k\neq i}} \sum_s \omega_{rs} \mu_{j\leftarrow k}^s. \tag{4.33}$$

In terms of these messages

$$q_i^r = \frac{\pi_r}{Z_i} \prod_j \Bigl(1 - \sum_s q_j^s \omega_{rs}\Bigr) \prod_{j\in\mathcal{N}_i} \sum_s \omega_{rs} \mu_{i\leftarrow j}^s, \tag{4.34}$$

while the normalizing factors Z_i and $Z_{i\leftarrow j}$ are chosen so that $\sum_r q_i^r = 1$ and $\sum_r \mu_{i\leftarrow j}^r = 1$ for all i, j.

Equation (4.33) is solved by simple iteration, starting for instance from random values, then the results are substituted into Eq. (4.34) to calculate q_i^r. Figure 11 shows an example application to a small network with two communities. The colours of the nodes indicate the probability that the node belongs to group 1 (red) or group 2 (blue). As we can see, the method has ably found the two groups of nodes in the network, with only a few nodes incorrectly assigned.

Message passing provides an efficient and practical algorithm for community detection in networks, but perhaps the most interesting result to come out of this approach is a more formal one. Like the other message passing applications we have seen, the equations for community detection can be used as a starting point for further calculations, particularly focusing on phase transition behaviour. Decelle *et al.* [32] considered what happens when one applies the message

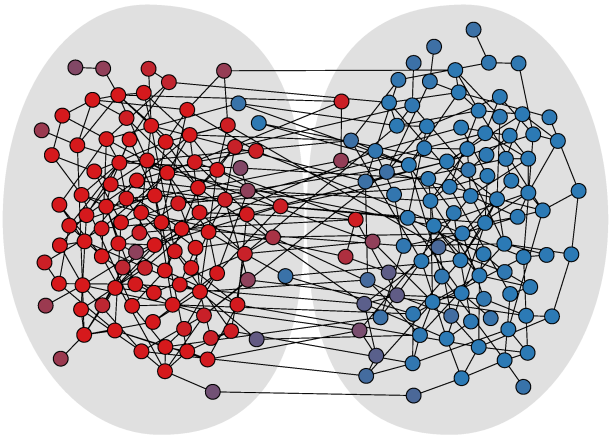

Figure 11. A small network with two communities, indicated by the shaded regions. The colours of the nodes correspond to the values of the message passing probabilities q_i^r from Eq. (4.34) and in this case the message passing algorithm has successfully found the two communities, with only a few nodes incorrectly assigned.

passing method to a network that was itself generated from the stochastic block model. In the simplest case, suppose we generate a network with an even number n of nodes divided into two equally sized groups, and suppose the edge probabilities ω_{rs} take just two values, ω_{in} when $r = s$ (within-group edges) and ω_{out} when $r \neq s$ (between-group edges).

Decelle *et al.* observe that in this situation the message passing equations, Eq. (4.33), have a symmetric solution $\mu^r_{i\leftarrow j} = \frac{1}{2}$ for all i, j, r, which also implies $q_i^r = \frac{1}{2}$. If the message passing iteration converges to this solution then it has failed to find any community structure in the network—every node is identically assigned, half-and-half, to both communities. Thus the algorithms fails if the symmetric solution is a stable fixed point of the iteration. Only if it is unstable can we find a nontrivial solution. As with our other message passing examples, it turns out that there is a bifurcation at which the fixed point changes from stable to unstable. This bifurcation is driven by changing values of the parameters ω_{in} and ω_{out}. As the values get close together the empirical distinction between within- and between-group edges becomes smaller and smaller, since the two have almost the same probability, and hence there is less and less signal of the communities embedded in the structure of the network. Beyond a certain point the signal becomes so weak that the algorithm fails to find anything, which is indicated by the symmetric fixed point becoming stable.

For the simple two-group example considered here, Decelle *et al.* showed that the transition falls at the point where

$$c_{\text{in}} - c_{\text{out}} = \sqrt{2(c_{\text{in}} + c_{\text{out}})}, \tag{4.35}$$

with $c_{\text{in}} = n\omega_{\text{in}}$ and $c_{\text{out}} = n\omega_{\text{out}}$. When the difference $c_{\text{in}} - c_{\text{out}}$ is smaller than this the algorithm fails to find the communities in the network. This transition point is known as the *detectability threshold* of the network.

But now we make a crucial observation. The calculation we have described is based on a fit of the stochastic block model to a network that was itself generated from the same model. But there is no better method for extracting the parameters of a network (or any data set) than fitting it to the model from which it was truly generated. This means that no other method for detecting the communities in this network can work better than the one described here. Hence if this method fails then *all* methods must fail. So all methods must fail below the detectability threshold.

This is a remarkable result. The question of whether there are communities in the network and where they are has a well-defined answer in this case: we know the communities in this artificial network because we put them in the network ourselves. And yet, provably, no algorithm

is able to find those communities. In other words, there are questions about networks for which there exist answers, and yet no method can ever find those answers. We might imagine, if we are clever enough and work hard enough, that we should be able to answer any question, but the work of Decelle *et al.* tells us this is not so. Some questions—even questions with well-defined answers—are unanswerable.

However, the result tells us more than this. This is not merely a statement about theoretical calculations and algorithms, but also about processes going on in the real world. Suppose there were some process taking place on a network, such as a social process or a biological process, whose outcome depended on whether there were communities in the network. Then that process would constitute an algorithm for detecting communities: it would give one outcome if there were communities present and another if there were not. Below the detectability threshold, however, no such algorithm can exist, which implies in turn that no such real-world process can exist either.

In a way, this is good news. It tells us that we only care about community structure when we are above the detectability threshold. If the structure in a network is undetectable then it can have no effect on any real-world outcome and so it doesn't matter whether it is present or not. Thus we only care about the easy cases of community detection, not the hard ones.

5. Loopy networks and correlated messages

In our discussion so far we have assumed that the neighbours j of a node i are not directly connected to one another by single edges or other short paths, or equivalently that our network contains no short loops. This assumption was necessary, for instance, to ensure that the probabilities of neighbours belonging to the giant cluster were independent in our percolation example, or that all walks that started along a particular edge returned along the same edge in our graph spectrum example.

A network that contains no loops at all is called a tree, and message passing methods are exact on trees. Message passing methods are not exact but typically work well on networks that have only long loops but no short ones. For instance, correlations between percolation probabilities on different nodes typically fall off exponentially with distance through the network, so the presence of long connecting paths introduces only small correlations and correspondingly small errors in our message passing results.

Many real networks, however, including especially social and biological networks, exhibit a high density of short loops, particularly triangles, the shortest loops of all. On such "loopy" networks the message passing methods we have described do not work well, giving poor approximations to the true answers for the problems they are designed to solve [34]. Indeed in some cases the message passing iteration may fail to converge at all on loopy networks, falling into limit cycles or chaotic trajectories instead. A number of efforts have been made to remedy these issues, with varying degrees of success [10,34,35]. In this section we describe some recent developments that offer a principled framework for bypassing the problem and creating message passing methods that work on loopy networks [36,37].

First, we need to make precise what we mean by a loop. A *cycle* in a network is a non-self-intersecting walk that starts and ends at the same node. "Non-self-intersecting" here means that no edge is traversed more than once in the walk. Consider Fig. 12a, which shows the neighbourhood of a certain node i in a network. Highlighted in blue are two cycles of length 3 starting (and ending) at i. Figure 12b shows two cycles of length 4 on the same network, highlighted in red and green. The latter two cycles, however, are qualitatively different from one another in an important way. The cycle in red follows edges that were already part of the cycles of length 3 in panel (a). If we know about the cycles of length 3 then, in a sense, we already know about this cycle of length 4 too. The cycle in green, however, is new in that it contains some edges we have not seen in any shorter cycle. We call this a *primitive cycle*. A primitive cycle from node i is a cycle starting and ending at i such that at least one of its edges is not contained in any shorter cycle from node i. This concept of the primitive cycle is central to understanding how to generalize message passing methods to loopy networks.

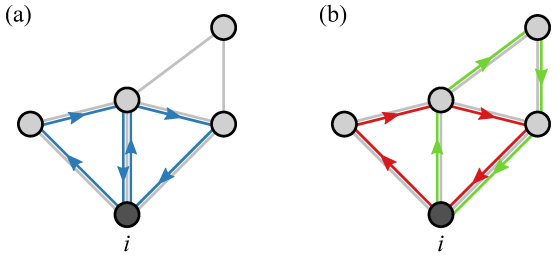

Figure 12. (a) Two cycles of length 3 in a small network, both starting and ending at the same node i. (b) Two cycles of length 4 in the same network. The cycle in green is a primitive cycle but the cycle in red is not because it is composed entirely of edges that were already present in the cycles of length 3.

Suppose we have a network that contains primitive cycles up to some length r only and no longer primitive cycles. It may contain longer non-primitive cycles—potentially many of them—but the longest primitive cycle, from any node anywhere in the network, has length r or less. For such a network we can write exact message passing equations according to the following recipe.

Around each node i in the network we construct a neighbourhood $\mathcal{N}_i$ that consists of the nodes and edges immediately adjacent to i plus all nodes and edges that lie on primitive cycles starting at i (of any length up to the network maximum of r). Figure 13 shows an example of such a neighbourhood in a network with maximum cycle length $r = 4$. For a network with $r = 2$, which is the smallest possible value, the neighbourhoods are the same as in our previous calculations, and the method reduces to standard message passing in this case. But for $r > 2$ the neighbourhoods may contain additional edges and nodes. Note that while, as we have said, there may also be non-primitive cycles starting from i, including ones of length greater than r, these are automatically contained within i's neighbourhood, since they are by definition made up entirely of edges that belong to primitive cycles of length r or less, which are themselves contained within the neighbourhood. Thus under this definition all cycles starting from i, of any length, are contained within the neighbourhood of i.

In our extended message passing scheme, node i receives messages $\mu_{i \leftarrow j}$ from all nodes j in $\mathcal{N}_i$. This means that some messages may come from nodes that are not directly connected to i via an edge. As in normal message passing the messages from neighbourhood nodes are themselves calculated from other messages received from outside the neighbourhood—see Fig. 13 again. But here a useful simplification occurs. Since the neighbourhood i includes all cycles of any length that start from node i, it follows that there are no paths connecting the nodes sending messages into the neighbourhood, other than through the neighbourhood itself. It there were, they would create cycles outside the neighbourhood, of which, by hypothesis, there are none. This observation is sufficient now to reestablish independence among the incoming messages and allow us to write message passing equations that work on these loopy networks.

This is not quite the whole story, however, for several reasons. First, as discussed at the start of this section, even in standard message passing we normally allow long loops in the network, since these introduce only small errors. We can do the same here. We stipulate that the network should contain no primitive cycles of length greater than r except "long" cycles, meaning ones long enough that the correlations they introduce are small in some sense. Second, it still remains to specify how we combine the messages coming into a node to calculate the node's properties, and this can be nontrivial. We will see an example in a moment. Third, real networks typically do not have a sharp cutoff in the length of their cycles as we have assumed, but instead have primitive cycles of all lengths, up to some (usually large) maximum. In practice, we approximate such networks by counting loops only up to some chosen length and ignoring longer loops. This turns out often to be an excellent approximation. Again we will see an example shortly.

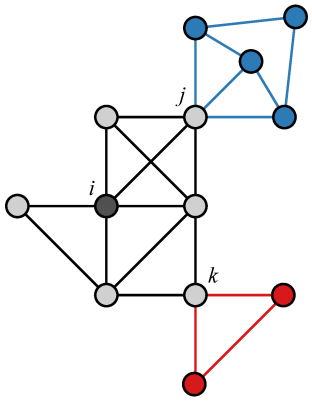

Figure 13. The neighbourhood $\mathcal{N}_i$ denoted by the nodes and edges in black and grey contains all primitive cycles of length four or less starting and ending at i. Node i receives messages from all other nodes in this neighbourhood, which themselves receive messages from their neighbourhoods. Node j, for instance, receives messages from the nodes in blue, but not from the grey nodes of $\mathcal{N}_i$, which are excluded as described in the text. We denote the blue neighbourhood by $\mathcal{N}_{i\leftarrow j}$. Note that, according to our definitions, $\mathcal{N}_{i\leftarrow j}$ necessarily meets $\mathcal{N}_i$ at only a single node—node j—and also that there cannot be any paths between pairs of neighbourhoods such as $\mathcal{N}_{i\leftarrow j}$ (in blue) and $\mathcal{N}_{i\leftarrow k}$ (in red), other than through $\mathcal{N}_i$ itself. If either of these conditions were violated then there would be primitive cycles of length longer than four starting from i.

(a) Example: Percolation

As an application of message passing on loopy networks let us look again at percolation with edge occupation probability p, and specifically at the calculation of the probability that a node in a network belongs to the giant percolation cluster. The first step in solving this problem is to construct the neighbourhood around each node, including all primitive cycles of length r or less, for some value of r that we choose. See Fig. 13 again for an example. Now each node j in the neighbourhood of i transmits a message $\mu_{i\leftarrow j}$ with value equal to the probability that node j is not connected to the giant cluster when i *and all of its neighbourhood* $\mathcal{N}_i$ (except for j itself) are removed from the network. As in Section 2, this removal ensures that we ignore any nodes j that are only connected to the giant cluster via $\mathcal{N}_i$.

The probability μ_i that i itself is not in the giant cluster is now equal to the probability that it is not connected to the giant cluster via any of the nodes j in $\mathcal{N}_i$. This probability still requires some effort to calculate. Figure 14 shows what is involved. The edges in the neighbourhood $\mathcal{N}_i$ may be occupied (with probability p) or not (probability $1-p$) and for every possible configuration Γ_i of occupied edges within the neighbourhood let $\sigma_{ij}(\Gamma_i)=1$ if there is at least one path of occupied edges from i to j within $\mathcal{N}_i$ and 0 otherwise. For instance, there is a path from i to j in Fig. 14, so $\sigma_{ij}=1$ for this configuration.

Now the total probability that i is not connected to the giant cluster is given by

$$\mu_i = \sum_{\Gamma_i} P[\Gamma_i] \prod_{j\in\mathcal{N}_i} \mu_{i\leftarrow j}^{\sigma_{ij}(\Gamma_i)}, \tag{5.1}$$

with $P[\Gamma_i]=p^m(1-p)^{k-m}$ being the probability of occurrence of the edge configuration Γ_i, where k is the number of edges in the neighbourhood and m is the number that are occupied in configuration Γ_i. Note how the product in Eq. (5.1) computes the total probability that none of the nodes j to which i is connected belong to the giant cluster and the sum averages this quantity over the probability distribution of configurations Γ_i.

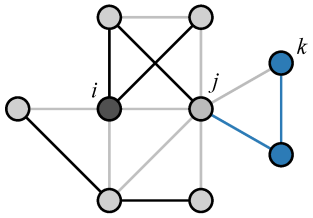

Figure 14. The bold edges in this diagram represent ones that are occupied, and for any configuration Γ_i of occupied edges in the neighbourhood $\mathcal{N}_i$ of node i the probability that i is connected to the giant cluster depends on the probabilities for each of the nodes j in $\mathcal{N}_i$ that are reachable along an occupied path. Similarly, the probability that node j is connected to the giant cluster in turn depends on the probabilities for each of its reachable neighbours k in $\mathcal{N}_{i\leftarrow j}$.

There are 2^k possible configurations of the k edges in the neighbourhood and hence naive evaluation of the sum in (5.1) takes time exponential in the neighbourhood size. For small values of the cycle length r the neighbourhoods will also be small, and hence the sum may be tractable. For larger values it may be necessary to approximate it. A convenient way of doing this is by Monte Carlo sampling of configurations. It turns out that one need only sample a small number of configurations to get accurate answers: ten or so is often sufficient. The reason is that one normally samples similar numbers of configurations for every neighbourhood in the network, and the effective number of configurations of the whole network that get sampled is the product of the numbers in each neighbourhood. If one samples ten configurations in the neighbourhood of each node then one is effectively sampling 10^n configurations of the entire network, which is typically a very large number and more than sufficient to give good convergence of the results.

We still need to calculate the values of the messages themselves, which can be done by a similar method. Since $\mu_{i\leftarrow j}$ depends only on connections outside $\mathcal{N}_i$, we define a modified neighbourhood $\mathcal{N}_{i\leftarrow j}$, which is the normal neighbourhood $\mathcal{N}_j$ with cycles up to length r, minus all nodes and edges in $\mathcal{N}_i$. Then

$$\mu_{i\leftarrow j} = \sum_{\Gamma_{i\leftarrow j}} P[\Gamma_{i\leftarrow j}] \prod_{k\in\mathcal{N}_{i\leftarrow j}} \mu_{j\leftarrow k}^{\sigma_{jk}(\Gamma_{i\leftarrow j})}, \tag{5.2}$$

where $\Gamma_{i\leftarrow j}$ is a configuration of occupied edges in $\mathcal{N}_{i\leftarrow j}$ and $\sigma_{jk}(\Gamma_{i\leftarrow j}) = 1$ if there is at least one occupied path from j to k in $\Gamma_{i\leftarrow j}$ and 0 otherwise.

Equation (5.2) we solve in the standard manner by iteration from any suitable set of starting values, then the converged values are used in (5.1) to calculate μ_i. From this probability we can also calculate other quantities of interest, such as the expected size S of the giant cluster, which is given as before by Eq. (2.6).

Figure 15 shows an example calculation performed on a real-world network, a social network of 13 861 nodes with a high density of triangles and other short loops. The plot shows the size of the giant percolation cluster as a function of the edge occupation probability p, calculated in three ways. The circles are the results of high-precision direct Monte Carlo simulations (not using message passing), which are laborious to perform but should give an accurate ground-truth result for comparison. The dashed line is calculated using the standard message-passing method of Eq. (2.5), which does not account for loops, and which does reasonably well in this case but shows clear deviations from the ground truth. The solid line shows the method of this section, accounting for loops of length up to four, and, as we can see, this calculation now agrees with the ground truth to high accuracy.

It is an open question what the maximum cycle length is that should be incorporated in a calculation like this in order to get good results. In our work we have found that one can truncate the calculation at surprisingly small cycle lengths—three, four or five—and still get excellent

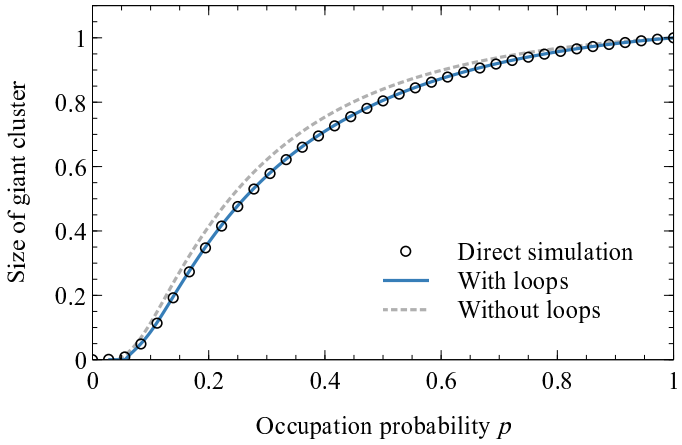

Figure 15. Size of the giant cluster for percolation on a large social network of 13 861 nodes. The points are results from direct simulation calculations and should be indicative of the ground truth in this case. The solid curve shows the results of a message passing calculation using Eqs. (5.1) and (5.2), including all cycles up to length four. The dashed line shows what happens when cycles are neglected. After Cantwell and Newman [36].

results, but the exact rate of convergence and the size of the errors introduced are presumably a function of the network structure and the nature of the dependence is not well understood.

The methods described here can also be extended to other message passing calculations on loopy networks, including the calculation of matrix spectra and the solution of spin models and other probabilistic models [36,37]. As an example, Fig. 16 shows a calculation of the spectral density of the adjacency matrix of a large software network—a PGP trust network—and again the message passing calculation compares favourably with the ground truth. It can also be significantly faster than traditional numerical methods, putting calculations for larger networks within reach: spectra for networks of over 300 000 nodes have been calculated using these methods on standard (non-parallel) hardware [36], something that would be impossible using traditional methods.

6. Conclusions

In this paper we have examined the use of message passing methods for the calculation of node properties on networks. These methods work by expressing the properties of each node in terms of those of its neighbours, leading to a set of self-consistent equations that are solved by numerical iteration. This approach has a number of advantages over more conventional numerical methods, being computationally efficient, especially on sparse networks, and allowing one to compute ensemble averages in a single calculation rather than by averaging over repeated simulations. We have given example applications of message passing to the calculation of percolation properties of networks, the evaluation of graph spectra, community detection in networks and the solution of thermal spin models such as the Ising model.

In addition to its use for numerical methods, we have also described how message passing forms the foundation for further analytic calculations, particularly of phase transition properties. One can regard the iteration of the message passing equations as a discrete-time dynamical system, whose bifurcations correspond to phase transitions of the original system under study. Examples include the percolation transition in percolation models, the ferromagnetic transition in the Ising model and the detectability threshold for community detection.

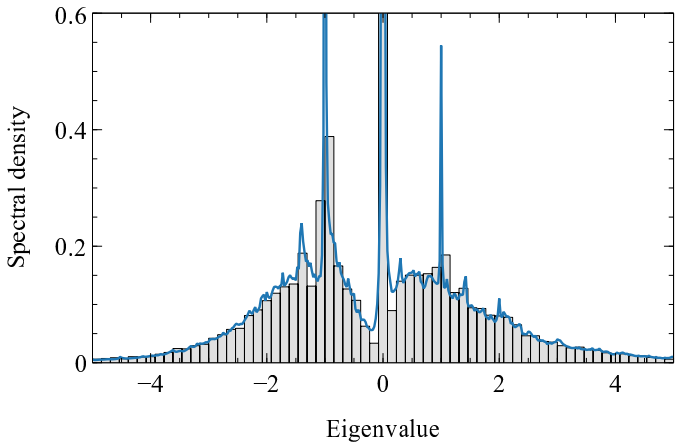

Figure 16. The adjacency matrix spectrum of a 10 680-node PGP network, a combination social/software network of trust relations between PGP keys and their owners. The histogram shows the distribution of eigenvalues from direct calculations using standard numerical methods, while the curve shows the spectral density from message passing including loops of length up to three. After Cantwell and Newman [36].

Finally, we have discussed a shortcoming of the message passing method, at least as it is traditionally formulated, that it works poorly on networks containing short loops because of loss of independence between the states of nearby nodes. We have reviewed recent work that provides a way around this shortcoming by defining messages that pass not only between immediately adjacent nodes but also between nodes within larger neighbourhoods that completely enclose the problematic loops. This approach leads to more complicated message passing equations but appears to give excellent results in example applications such as percolation and the calculation of matrix spectra.

Acknowledgements

The author thanks George Cantwell, Alec Kirkley, Cris Moore and Austin Polanco for helpful conversations. This work was funded in part by the US National Science Foundation under grants DMS–1710848 and DMS–2005899.

References

1. Strogatz SH. 2001 Exploring complex networks. *Nature* **410**, 268–276.
2. Watts DJ. 2003 *Six Degrees: The Science of a Connected Age*. New York: Norton.
3. Boccaletti S, Latora V, Moreno Y, Chavez M, Hwang DU. 2006 Complex networks: Structure and dynamics. *Physics Reports* **424**, 175–308.
4. Barabási AL. 2016 *Network Science*. Cambridge: Cambridge University Press.
5. Newman M. 2018 *Networks*. Oxford: Oxford University Press 2 edition.
6. Bethe HA. 1935 Statistical theory of superlattices. *Proc. R. Soc. London A* **150**, 552–575.
7. Pearl J. 1982 Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the 2nd National Conference on Artificial Intelligence* pp. 133–136 Palo Alto, CA. AAAI Press.
8. Mézard M, Montanari A. 2009 *Information, Physics, and Computation*. Oxford: Oxford University Press.
9. Mézard M, Parisi G, Zecchina R. 2002 Analytic and algorithmic solution of random satisfiability problems. *Science* **297**, 812–815.

10. Yedidia JS, Freeman WT, Weiss Y. 2003 Understanding belief propagation and its generalizations. In Lakemeyer G, Nebel B, editors, *Exploring Artificial Intelligence in the New Millennium* pp. 239–270. San Francisco, CA: Morgan Kaufmann.
11. Albert R, Jeong H, Barabási AL. 2000 Attack and error tolerance of complex networks. *Nature* **406**, 378–382.
12. Cohen R, Erez K, ben-Avraham D, Havlin S. 2000 Resilience of the Internet to random breakdowns. *Phys. Rev. Lett.* **85**, 4626–4628.
13. Karrer B, Newman MEJ, Zdeborová L. 2014 Percolation on sparse networks. *Phys. Rev. Lett.* **113**, 208702.
14. Hamilton KE, Pryadko LP. 2014 Tight lower bound for percolation threshold on a infinite graph. *Phys. Rev. Lett.* **113**, 208701.
15. Krzakala F, Moore C, Mossel E, Neeman J, Sly A, Zdeborová L, Zhang P. 2013 Spectral redemption in clustering sparse networks. *Proc. Natl. Acad. Sci. USA* **110**, 20935–20940.
16. Martin T, Zhang X, Newman MEJ. 2014 Localization and centrality in networks. *Phys. Rev. E* **90**, 052808.
17. Yoon S, Goltsev AV, Dorogovtsev SN, Mendes JFF. 2011 Belief-propagation algorithm and the Ising model on networks with arbitrary distributions of motifs. *Phys. Rev. E* **84**, 041144.
18. Bonacich PF. 1987 Power and centrality: A family of measures. *Am. J. Sociol.* **92**, 1170–1182.
19. Porter MA, Gleeson J. 2016 *Dynamical Systems on Networks: A Tutorial*. Berlin: Springer.
20. Fortunato S. 2010 Community detection in graphs. *Phys. Rep.* **486**, 75–174.
21. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. 1992 *Numerical Recipes in C*. Cambridge: Cambridge University Press.
22. Rogers T, Pérez Castillo I, Kühn R, Takeda K. 2008 Cavity approach to the spectral density of sparse symmetric random matrices. *Phys. Rev. E* **78**, 031116.
23. Newman MEJ, Zhang X, Nadakuditi RR. 2019 Spectra of random networks with arbitrary degrees. *Phys. Rev. E* **99**, 042309.
24. McKay BD. 1981 The expected eigenvalue distribution of a large regular graph. *Linear Algebra Appl.* **40**, 203–216.
25. Newman MEJ, Martin T. 2014 Equitable random graphs. *Phys. Rev. E* **90**, 052824.
26. Molloy M, Reed B. 1995 A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms* **6**, 161–179.
27. Newman MEJ, Strogatz SH, Watts DJ. 2001 Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E* **64**, 026118.
28. Holland PW, Laskey KB, Leinhardt S. 1983 Stochastic blockmodels: First steps. *Social Networks* **5**, 109–137.
29. Airoldi EM, Blei DM, Fienberg SE, Xing EP. 2008 Mixed membership stochastic blockmodels. *Journal of Machine Learning Research* **9**, 1981–2014.
30. Karrer B, Newman MEJ. 2011 Stochastic blockmodels and community structure in networks. *Phys. Rev. E* **83**, 016107.
31. Peixoto TP. 2017 Nonparametric Bayesian inference of the microcanonical stochastic block model. *Phys. Rev. E* **95**, 012317.
32. Decelle A, Krzakala F, Moore C, Zdeborová L. 2011 Inference and phase transitions in the detection of modules in sparse networks. *Phys. Rev. Lett.* **107**, 065701.
33. Moore C. 2017 The computer science and physics of community detection: Landscapes, phase transitions, and hardness. Preprint arxiv:1702.00467.
34. Frey BJ, MacKay DJC. 1998 A revolution: Belief propagation in graphs with cycles. In Jordan MI, Kearns MJ, Solla SA, editors, *Proceedings of the 1997 Conference on Neural Information Processing Systems* pp. 479–485 Cambridge, MA. MIT Press.
35. Chertkov M, Chernyak VY. 2006 Loop calculus in statistical physics and information science. *Phys. Rev. E* **73**, 065102.
36. Cantwell GT, Newman MEJ. 2019 Message passing on networks with loops. *Proc. Natl. Acad. Sci. USA* **116**, 23398–23403.
37. Kirkley A, Cantwell GT, Newman MEJ. 2021 Belief propagation for networks with loops. *Science Advances* **7**, eabf1211.