Supplementary Information for

Message passing on networks with loops

George T. Cantwell and M. E. J. Newman

This PDF file includes:

Supplementary text
References for SI reference citations

Supporting Information Text

Derivation of the message passing equations

Below we provide some additional details of the derivation of the fundamental message passing equations, Eqs. (4) and (19) in the paper.

Percolation. The derivation of Eq. (4) in the paper follows similar lines to that of Eq. (3). By analogy with Eq. (2) we can write a generating function for $\pi_{i\leftarrow j}(s|\Gamma_{j\setminus i})$ thus:

$$\begin{aligned} H_{i\leftarrow j}(z|\Gamma_{j\setminus i}) &= \sum_s \pi_{i\leftarrow j}(s|\Gamma_{j\setminus i})\, z^s = \sum_s z^s \Biggl\{ \sum_{\{s_k:k\in\Gamma_{j\setminus i}\}} \Biggl[\prod_{k\in\Gamma_{j\setminus i}} \pi_{j\leftarrow k}(s_k) \Biggr] \delta(s-1, \textstyle\sum_{k\in\Gamma_{j\setminus i}} s_k) \Biggr\} \\ &= z \prod_{k\in\Gamma_{j\setminus i}} \sum_{s_k} z^{s_k} \pi_{j\leftarrow k}(s_k) = z \prod_{k\in\Gamma_{j\setminus i}} H_{j\leftarrow k}(z) = z \prod_{j\in N^{(r)}_{j\setminus i}} [H_{j\leftarrow k}(z)]^{w_{j\setminus i,k}}, \end{aligned} \tag{A.1}$$

where in the last line we have introduced the random variable $w_{j\setminus i,k}$ which takes the value 1 if $k \in \Gamma_{j\setminus i}$ and 0 otherwise. In other words, $w_{j\setminus i,k} = 1$ if there is a path of occupied edges from j to k in $N^{(r)}_{j\setminus i}$. To compute the generating function for $\pi_{i\leftarrow j}(s)$ we simply average Eq. (A.1) over the possible realizations of $\Gamma_{j\setminus i}$, which leads to the message passing equations

$$\begin{aligned} H_{i\leftarrow j}(z) &= \sum_s \pi_{i\leftarrow j}(s)\, z^s = \Bigl\langle \sum_s \pi_{i\leftarrow j}(s|\Gamma_{j\setminus i})\, z^s \Bigr\rangle_{\Gamma_{j\setminus i}} = z \Bigl\langle \prod_{k\in N^{(r)}_{j\setminus i}} H_{j\leftarrow k}(z)^{w_{j\setminus i,k}} \Bigr\rangle_{\Gamma_{j\setminus i}} \\ &= zG_{i\leftarrow j}\bigl(\mathbf{H}_{j\leftarrow}(z)\bigr), \end{aligned} \tag{A.2}$$

as stated in the paper.

Spectrum. The derivation of the message passing equations for matrix spectra is more complex than for percolation and is given only in abbreviated form in the paper. Here we give the full derivation including intermediate algebraic steps.

As described in the paper, the spectral density of a symmetric matrix $\mathbf{A}$ is given by

$$\rho(z) = -\frac{1}{n\pi z} \sum_{s=0}^{\infty} \sum_{i=1}^{n} \frac{X_i^s}{z^s}, \tag{A.3}$$

where X_i^s is the sum of the weights of all closed walks of length s that start and end at node i. This sum can be expressed in terms of the sum Y_i^s of the weights of all *excursions* of length s by Eq. (10) in the paper, which we repeat here for convenience:

$$X_i^s = \sum_{m=0}^{\infty} \Biggl[\sum_{s_1=1}^{\infty} \cdots \sum_{s_m=1}^{\infty} \delta\bigl(s, \textstyle\sum_{u=1}^m s_u\bigr) \displaystyle\prod_{u=1}^{m} Y_i^{s_u} \Biggr]. \tag{A.4}$$

Substituting this expression into Eq. (A.3) we get

$$\rho(z) = -\frac{1}{n\pi z} \sum_{i=1}^{n} \sum_{m=0}^{\infty} \prod_{u=1}^{m} \Biggl[\sum_{s=1}^{\infty} \frac{Y_i^s}{z^s} \Biggr], \tag{A.5}$$

and, defining the function

$$H_i(z) = \sum_{s=1}^{\infty} \frac{Y_i^s}{z^{s-1}}, \tag{A.6}$$

we find that

$$\rho(z) = -\frac{1}{n\pi z} \sum_{i=1}^{n} \sum_{m=0}^{\infty} \biggl[\frac{H_i(z)}{z} \biggr]^m = -\frac{1}{n\pi} \sum_{i=1}^{n} \frac{1}{z - H_i(z)}, \tag{A.7}$$

as stated in the paper.

The function $H_i(z)$ we calculate from Eq. (15), which tells us that

$$H_i(z) = \sum_{l=0}^{\infty} \frac{1}{z^l} \sum_{w\in W_i^l} |w| \prod_{j\in w} \sum_{m=0}^{\infty} \prod_{k=1}^{m} \sum_{s=1}^{\infty} \frac{Y_{i\leftarrow j}^s}{z^s} = \sum_{w\in W_i} |w| \prod_{j\in w} \frac{1}{z - H_{i\leftarrow j}(z)}, \tag{A.8}$$

where W_i is the set of excursions of all lengths in the neighborhood of i, $|w|$ is the weight of excursion w (i.e., the product of the matrix elements along the excursion), and

$$H_{i\leftarrow j}(z) = \sum_{s=1}^{\infty} \frac{Y_{i\leftarrow j}^s}{z^{s-1}}. \tag{A.9}$$

By an equivalent line of argument we can also show that

$$H_{i\leftarrow j}(z) = \sum_{w\in W_{j\setminus i}} |w| \prod_{k\in w} \frac{1}{z - H_{j\leftarrow k}(z)}. \tag{A.10}$$

This last expression defines the message passing equations for the spectral density calculation. For any given value of z they can be iterated to calculate the spectral density via Eqs. (A.7) and (A.8).

As discussed in the paper, the efficiency of this approach relies crucially on being able to perform the sum over excursions w from node j efficiently, which we do as follows. If excursion w returns to j after just a single step (via a self-loop) then it has weight $|w| = A_{jj}$. Otherwise, if it takes two or more steps for a total of $l+1$ steps, visiting l (not necessarily distinct) nodes $k_1, k_2, \ldots, k_l$ along the way (other than the starting node), then the weight is

$$|w| = A_{j,k_1} \left(\prod_{m=1}^{l-1} A_{k_m,k_{m+1}} \right) A_{k_l,j}. \tag{A.11}$$

Inserting these values into Eq. (A.10) we get

$$H_{i\leftarrow j}(z) = A_{jj} + \sum_{l=1}^{\infty} \sum_{w\in W_{j\setminus i}^l} \frac{A_{j,k_1}}{z - H_{j\leftarrow k_1}(z)} \left(\prod_{m=1}^{l-1} \frac{A_{k_m,k_{m+1}}}{z - H_{j\leftarrow k_{m+1}}} \right) A_{k_l,j} \tag{A.12}$$

where $W_{j\setminus i}^l$ is the set of all excursions of length $l+1$ in $N_{j\setminus i}$. The sum over excursions is equivalent to a sum over all possible sets of l nodes $k_1 \ldots k_l$ within the neighborhood, so we can write

$$H_{i\leftarrow j}(z) = A_{jj} + \sum_{l=1}^{\infty} \sum_{k_1} \cdots \sum_{k_l} \frac{A_{j,k_1}}{z - H_{j\leftarrow k_1}(z)} \left(\prod_{m=1}^{l-1} \frac{A_{k_m,k_{m+1}}}{z - H_{j\leftarrow k_{m+1}}} \right) A_{k_l,j}. \tag{A.13}$$

Defining $\mathbf{v}_{i\leftarrow j}$ to be the vector with elements $v_{i\leftarrow j,k} = A_{jk}$ if nodes j and k are directly connected in $N_{j\setminus i}^{(r)}$ and 0 otherwise, $\mathbf{A}^{i\leftarrow j}$ to be the matrix for the neighborhood of j with the neighborhood of i removed, such that

$$A_{kl}^{i\leftarrow j} = \begin{cases} A_{kl} & \text{for } k,l \neq j \text{ and edge } (k,l) \in N_{j\setminus i}^{(r)}, \\ 0 & \text{otherwise,} \end{cases} \tag{A.14}$$

and $\mathbf{D}^{i\leftarrow j}(z)$ to be the diagonal matrix with entries $D_{kk}^{i\leftarrow j} = z - H_{j\leftarrow k}(z)$, we then have

$$\begin{aligned} H_{i\leftarrow j}(z) &= A_{jj} + \sum_{l=1}^{\infty} \sum_{k_1} \sum_{k_l} v_{i\leftarrow j,k_1} \left(D_{k_1,k_1}^{i\leftarrow j} \right)^{-1} \left[\mathbf{A}^{i\leftarrow j} \left(\mathbf{D}^{i\leftarrow j} \right)^{-1} \right]_{k_1,k_l}^{l-1} v_{i\leftarrow j,k_l} \\ &= A_{jj} + \left[\left(\mathbf{D}^{i\leftarrow j} \right)^{-1} \mathbf{v}_{i\leftarrow j} \right]^T \left[\mathbf{I} - \mathbf{A}^{i\leftarrow j} \left(\mathbf{D}^{i\leftarrow j} \right)^{-1} \right]^{-1} \mathbf{v}_{i\leftarrow j} \\ &= A_{jj} + \mathbf{v}_{i\leftarrow j}^T \left(\mathbf{D}^{i\leftarrow j} - \mathbf{A}^{i\leftarrow j} \right)^{-1} \mathbf{v}_{i\leftarrow j}, \end{aligned} \tag{A.15}$$

as stated in the paper.

Monte Carlo algorithm for $G_i(\mathbf{y})$

In the message passing equations for bond percolation, Eqs. (3) and (4) in the paper, the quantity $G_i(\mathbf{y})$ is a generating function encoding the probability that we can reach nodes in the neighborhood $N_i^{(r)}$ of a given node i by following occupied edges. It is defined by

$$G_i(\mathbf{y}) = \Big\langle \prod_{j\in N_i^{(r)}} y_j^{w_{ij}} \Big\rangle_{\Gamma_i}, \tag{B.1}$$

where w_{ij} is a binary (zero/one) random variable indicating whether node j is reachable from node i and the average is performed over all possible sets Γ_i of reachable nodes, each weighted by the sum of the probabilities of all edge configurations that can give rise to that particular set. The number of such configurations can become large as the size of the neighborhood grows, making exhaustive averages difficult to perform numerically. For larger neighborhoods, therefore, we employ a Monte Carlo averaging scheme as follows.

Suppose that node i has degree k_i and that there are $k_i + M$ edges in the neighborhood $N_i^{(r)}$, with k_i of them directly connected to i and M additional edges that complete cycles between i's neighbors. For locally tree-like networks there are no cycles and $M = 0$, but in general $M \geq 0$. Let $G_i(\mathbf{y}|m)$ be the value of $G_i(\mathbf{y})$ when exactly m of the M additional edges are occupied, which happens with probability $\binom{M}{m}p^m(1-p)^{M-m}$. Then we can write $G_i(\mathbf{y})$ itself in the form

$$G_i(\mathbf{y}) = \sum_{m=0}^{M} G_i(\mathbf{y}|m)\binom{M}{m}p^m(1-p)^{M-m}. \tag{B.2}$$

Our algorithm works by making a Monte Carlo estimate of $G_i(\mathbf{y}|m)$ using a version of the algorithm of Newman and Ziff (1) and then applying Eq. (B.2). The basic idea is to occupy edges one by one and keep track of the connected percolation clusters using an efficient union-find data structure based on pointers (1). Using this data structure the algorithm is able to determine whether two nodes belong to the same cluster, or to join two clusters together, in (very nearly) constant time. To compute $G_i(\mathbf{y}|m)$ itself, the algorithm maintains a record of two quantities for each cluster, a real value x and a probability q. In detail the algorithm works as follows.

The clusters we consider are the sets of nodes in the neighborhood, other than i, that are connected via occupied edges in $N_i(r)$ but not via node i itself, i.e., via the M additional edges mentioned above. Initially none of the M edges is occupied and each node is a cluster in its own right. For each of these one-node clusters j we assign $x_j = y_j$ and we set $q_j = 1 - p$ if node j is a direct neighbor of i or $q_j = 1$ otherwise. We also compute the quantity

$$u_0 = \prod_j \left(q_j + (1-q_j)\,x_j\right). \tag{B.3}$$

Now we occupy the M edges one by one in random order. Let j_1 and j_2 be the nodes at the ends of the mth edge occupied. If j_1 and j_2 are already part of the same cluster before the edge is added (which, as we have said, we can determine in time O(1)), then we set

$$u_m \leftarrow u_{m-1}. \tag{B.4}$$

Otherwise, if j_1 and j_2 are in different clusters r and s, then the addition of the mth edge joins r and s together (which again we can achieve in O(1) time) to make a larger cluster which, without loss of generality, we will label r. At the same time we set

$$u_m \leftarrow \frac{u_{m-1}}{[q_r + (1-q_r)\,x_r][q_s + (1-q_s)x_s]}, \tag{B.5}$$

$$x_r \leftarrow x_r x_s, \tag{B.6}$$

$$q_r \leftarrow q_r q_s, \tag{B.7}$$

$$u_m \leftarrow u_m\big[q_r + (1-q_r)x_r\big]. \tag{B.8}$$

After all M edges have been occupied, the $M+1$ quantities u_m with $m = 0 \ldots M$ give us an estimate of $G_i(\mathbf{y}|m)$, and $G_i(\mathbf{y})$ can be calculated from Eq. (B.2) as

$$G_i(\mathbf{y}) \simeq \sum_{m=0}^{\infty} u_m \binom{M}{m} p^m(1-p)^{M-m}. \tag{B.9}$$

The calculation of $G_{i\leftarrow j}(\mathbf{y})$ is identical except for the replacement of the neighborhood by $N_{j\setminus i}^{(r)}$. Finally, we average the results over repeated runs of the algorithm to get our estimate of the generating functions. We find surprisingly good results with averages over a relatively small number of runs—we used just eight runs for each neighborhood to generate the results shown in Fig. (2).

Note that the sequence of edges added and cluster joins performed does not depend on the values of either $\mathbf{y}$ or p, which means we can use the same sequence to calculate $G_i(\mathbf{y})$ for many different $\mathbf{y}$ and p. We can also use the same sequence on successive iterations of the message passing process, which has the benefit of removing any statistical fluctuations between iterations and is useful when estimating convergence of the message passing process, which can otherwise be difficult to do.

As is often the case for Monte Carlo calculations, it is not easy to say exactly how many runs will be required to get good results. Note, however, that if we perform S runs for each neighborhood then, because neighborhoods are sampled independently, we effectively generate S^n configurations of the whole network, and this number can become very large for large n even when S is small. Thus we expect to get good answers even with quite modest values of S, and indeed this is what we see in the calculations reported in the paper.

Other applications

An important application of message passing algorithms within physics is for the calculation of partition functions. Consider for example a spin model such as the Ising model on a network. On a tree one can show that the probability of any given spin state $\boldsymbol{\sigma}$ factorizes exactly according to

$$P(\boldsymbol{\sigma}) = \frac{\prod_{(i,j)} P(\sigma_i, \sigma_j)}{\prod_i P(\sigma_i)^{k_i - 1}}, \tag{C.1}$$

where $P(\sigma_i)$ is the marginal distribution for the spin on node i and $P(\sigma_i, \sigma_j)$ is the joint marginal distribution for the spins on nodes i and j. The product in the numerator is over edges (i,j) in the network and k_i is the degree of node i. For the Ising model one can write

$$P(\sigma_i) = \frac{1}{Z_i} e^{-\beta h_i \sigma_i}, \qquad P(\sigma_i, \sigma_j) = \frac{1}{Z_{ij}} e^{-\beta(\sigma_i\sigma_j + h_{ji}\sigma_i + h_{ij}\sigma_j)}, \tag{C.2}$$

where h_i and h_{ij} are effective fields felt by the spins, which can be calculated using a message passing method (2), and the normalizing constants Z_i and Z_{ij} can be computed by enforcing $\sum_{\sigma_i} P(\sigma_i) = 1$ and $\sum_{\sigma_i,\sigma_j} P(\sigma_i, \sigma_j) = 1$. Once we know Z_i and Z_{ij} we can calculate the complete partition function $\mathcal{Z}$ from Eq. (C.1), which gives

$$\ln \mathcal{Z} = \sum_i (k_i - 1) \ln Z_i - \sum_{(i,j)} \ln Z_{ij}. \tag{C.3}$$

This equation is exact only if the network is a tree. When applied to non-trees it is known as the Bethe approximation.

An analogous computation can be performed for networks containing short loops using the concepts we have developed. In that case the distribution over spin states would be factorized over neighborhoods,

$$P(\boldsymbol{\sigma}) = \frac{\prod_{(i,j)} P\left(\{\sigma_k | k \in N_i \cup N_j\}\right)}{\prod_i P\left(\{\sigma_k | k \in N_i\}\right)^{|N_i| - 1}}, \tag{C.4}$$

where now the product in the numerator is over all pairs of nodes (i,j) that are in each other's neighborhoods and $|N_i|$ is the number of nodes in N_i, not including i itself. The equivalent of the Bethe approximation, Eq. (C.3), is then

$$\ln \mathcal{Z} = \sum_i (|N_i| - 1) \ln Z_i - \tfrac{1}{2} \sum_i \sum_{j \in N_i} \ln Z_{ij}. \tag{C.5}$$

We leave the thorough exploration of this approximation for future work.

References

1. M. E. J. Newman and R. M. Ziff, Efficient Monte Carlo algorithm and high-precision results for percolation. *Phys. Rev. Lett.* **85**, 4104–4107 (2000).
2. M. Mézard and G. Parisi, The Bethe lattice spin glass revisited. *Eur. Phys. J. B* **20**, 217–233 (2001).