PHYSICAL REVIEW E **74**, 035102(R) (2006)

Community detection as an inference problem

M. B. Hastings
Center for Nonlinear Studies and Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA
(Received 21 April 2006; published 15 September 2006)

We express community detection as an inference problem of determining the most likely arrangement of communities. We then apply belief propagation and mean-field theory to this problem, and show that this leads to fast, accurate algorithms for community detection.

DOI: 10.1103/PhysRevE.74.035102 PACS number(s): 89.75.Hc, 05.10.−a, 89.70.+c

Community detection is a well-studied problem in networks [1]. This is the problem of dividing a network into communities, such that nodes within the same community tend to be connected by links, while those within different communities tend not to be connected by links. This problem has applications in understanding the structure of social and biological networks [2], while the closely related graph partitioning problem discussed below has applications in parallel processing, to allocate assignments to different processors while minimizing interprocessor communication.

Unfortunately, despite all this interest, there is no formal definition of the problem. Instead, each author tends to define communities as being whatever is found by a particular community detection algorithm [3]. In this work, we exploit a standard method of testing communities, the four-groups test [4], to express community detection as an inference, or maximum likelihood problem. This leads to a derivation of a Potts model similar to those derived previously on phenomenological grounds [5].

To solve this inference problem, we must find the ground state of the Potts model. To do this, we turn to the techniques of *belief propagation* [7], also known as sum-product and mean-field theory. Belief propagation was originally developed to perform decoding in a certain class of error correcting codes, called low-density parity check codes [8]. In this problem, one has a sender and receiver communicating over a noisy channel, and the receiver must determine which of all the possible messages is the most likely. This problem can be mapped onto finding the ground state of a spin system on a particular graph [9]. The belief propagation algorithm exploits the fact that the graph has a low density of loops to solve this problem, in a manner similar to the famous Bethe-Peierls [10] solution for the thermodynamics of a spin system on a Bethe lattice.

We will find that the resulting belief propagation algorithm for community detection is highly accurate on the four-groups problem, while also performing well on other test networks. We then discuss the scaling of computational time with system size, extensions of this algorithm, and other problems.

To express community detection as an inference problem we consider the following method, often used to test community detection algorithms [4]. We invent a network as follows: we consider N nodes, divided into q different communities with n_k, $k=1,\ldots,q$, nodes in each community. The initial assignment of nodes to communities is done randomly in one of two ways. There are two ways used in the literature: either we assign each node independently to a random community (leading, in the case $q=2$ to a binomial distribution of the number of nodes to a given community), or we randomly assign exactly N/q nodes to each community. We then consider each pair of nodes in turn. We connect those nodes with probability p_{in} if they lie within the same community and p_{out} if they lie within different communities. We then run the community detection algorithm and see if it correctly assigns nodes to communities. Let q_i be the initial assignment of a community to node i. The probability that a given graph G arises from this procedure is equal to

$$p(G|\{q_i\}) = \left(\prod_{k=1}^{q}(1-p_{in})^{n_k^2/2-n_k/2}\right)\left(\prod_{k=1}^{q-1}\prod_{l=k+1}^{q}(1-p_{out})^{n_k n_l}\right) \times \left(\prod_{\langle ij\rangle}\left(\frac{p_{in}}{1-p_{in}}\right)^{\delta_{q_i,q_j}}\right)\left(\prod_{\langle ij\rangle}\left(\frac{p_{out}}{1-p_{out}}\right)^{1-\delta_{q_i,q_j}}\right) \tag{1}$$

where $\Pi_{\langle ij\rangle}$ denotes a product over pairs of vertices i,j connected by an edge. To verify the correctness of this formula, first consider the case in which there are no edges at all in the graph. Then, the probability is given correctly by the first two products of Eq. (1). Adding edges to the graph changes the probability as given by the second two products in Eq. (1).

We now consider a given graph and formulate the problem as follows. Find the most likely community assignment. Following Bayes' theorem, given a graph, the probability that any given community assignment is the "correct" assignment is proportional to the probability $p(G|\{q_i\})$ multiplied by the *a priori* probability of having a given n_k:

$$P(\{q_i\}|G) = P(G|\{q_i\})P(\{q_i\})/P(G). \tag{2}$$

If we follow the procedure of randomly assigning each node to a given community, then the *a priori* probability $P(\{q_i\})$ is constant, and thus the optimal community assignment maximizes the probability in Eq. (1); any nonconstant prior probability such as arises by constraining there to be exactly N/q nodes in each community can be easily incorporated by adding additional terms. Note that $P(G)$ is unimportant in this optimization procedure as it does not depend on the specific choice of $\{q_i\}$ being only an overall proportionality constant for the given graph.

We rewrite Eq. (1) as an exponential:

$$p(G|\{q_i\}) = c \exp\left(\sum_{\langle ij\rangle} J\delta_{q_i,q_j}\right)\exp\left(\sum_{i\neq j} J'\delta_{q_i,q_j}/2\right), \quad (3)$$

where $\Sigma_{\langle ij\rangle}$ denotes a sum over pairs of i,j connected by an edge in the graph, and $c=\exp[\ln(1-p_{out})N(N-1)/2] \times\exp\{\Sigma_{\langle ij\rangle}\ln[p_{out}/(1-p_{out})]\}$, and with

$$J = \ln\{[p_{in}(1-p_{out})]/[(1-p_{in})p_{out}]\},$$

$$J' = \ln[(1-p_{in})/(1-p_{out})]. \quad (4)$$

The factor of 1/2 in Eq. (3) is to avoid double counting. Equation (3) presents the probability as a Potts model problem with combined short- and long-range interactions, with coupling constants J,J'. Thus, for the case of random assignment of nodes to communities, Eq. (3) gives the exact probability of any initial community assignment, reducing the problem of community detection to finding the ground state of this Potts model. Assuming $p_{in}>p_{out}$, the short-range interactions are ferromagnetic, favoring the assignment of neighboring nodes to the same community, while the long-range interactions are antiferromagnetic and prevent one from simply taking all nodes to lie within the same community. The problem of finding the ground state is very closely related to the NP-complete problem of graph partitioning, to break a graph up into partitions, minimizing the number of edges connecting partitions and minimizing the difference in number of nodes between partitions. Having arrived at Eq. (3) we have a very similar problem to that studied in [5]. Other problems in combining Potts models with clustering appear in [6].

Instead of using Monte Carlo methods to find the ground state as in [5], we adopt the method of belief propagation, which we believe to be more efficient for many of the community detection problems that arise. Indeed, for the inference problems which arise in many error correcting codes, belief propagation is the most efficient method.

We begin by taking a mean-field approximation for the long-range interactions, justified for large N. The reason for this will be discussed more below. We approximate $p(G|\{q_i\})\approx p_{mft}(\{q_i\})$, where

$$p_{mft}(\{q_i\}) \equiv Z^{-1}\exp\left(\sum_{\langle ij\rangle} J\delta_{q_i,q_j}\right)\exp\left(\sum_i h_i(q_i)\right). \quad (5)$$

Here Z is a normalization, and for $r=1,\ldots,q$ we define $h_i(r)=J'N\rho(r)-J'p_i(r)$ and $\rho(r)=\Sigma_i p_i(r)/N$, with

$$p_i(r) = \frac{\sum_{\{q_j\},q_i=r} p_{mft}(\{q_j\})}{\sum_{\{q_j\}} p_{mft}(\{q_j\})}, \quad (6)$$

so that $p_i(r)$ is the probability in the mean-field approximation that node i belongs to community r. These are a set of self-consistent equations for $p_i(q)$.

We will find that solving these equations, at least in the belief propagation approximation below, leads to a spontaneous symmetry breaking: for sufficiently large J,J', the probability $p_i(q)$ depends on q. Of course, given any solution that breaks symmetry, one can arrive at other valid solutions by relabeling the communities. We then make a "nodewise" maximum *a posteriori* probability approximation: for each node i, we compute $p_i(q)$ and then assign the node i to the community q that maximizes $p_i(q)$. This is an approximation: the set of community assignments that maximizes $p(G|\{q_i\})$ may not be given by maximizing the probability for each node separately. However, similar approximations work very well for error correcting codes [8], and these approximations are justified for large J,J'.

To compute the $p_i(q)$, we apply belief propagation. Suppose the graph forms a tree, with no loops. Then, the problem of solving for $p_i(q)$ given $h_i(q)$ can be solved: for each pair of nodes i,j connected by an edge, we define $p_{ij}(r)$ to be the probability p_i for the network modified by removing the edge connecting i to j. That is,

$$p_{ij}(r) = \frac{\sum_{\{q_k\},q_i=r} \exp(-J\delta_{q_i,q_j})p_{mft}(\{q_k\})}{\sum_{\{q_k\}} \exp(-J\delta_{q_i,q_j})p_{mft}(\{q_k\})}, \quad (7)$$

as $\exp(-J\delta_{q_i,q_j})p_{mft}(\{q_k\})$ is the probability distribution on the network with the edge removed. Then, for a treelike structure, the Bethe-Peierls solution gives

$$p_i(r) = Z_i^{-1}\exp[h_i(r)]\prod_j^{i,j\ \text{n.n.}} [\exp(J)p_{ji}(r) + (1-p_{ji}(r))], \quad (8)$$

where Z_i is chosen so that $\Sigma_{r=1}^q p_i(r)=1$, and where the product $\Pi_j^{i,j\ \text{n.n.}}$ is taken over all nodes j which are connected to node i. Note that $\exp(J)p_{ji}(r)+[1-p_{ji}(r)] =\Sigma_s p_{ji}(s)[\delta_{r,s}\exp(J)+(1-\delta_{r,s})]$. Similarly,

$$p_{ij}(r) = Z_{ij}^{-1}\exp[h_i(r)]\prod_{k\neq j}^{i,k\ \text{n.n.}} [\exp(J)p_{ki}(r) + (1-p_{ki}(r))], \quad (9)$$

where again we choose Z_{ij} such that $\Sigma_{r=1}^q p_{ij}(r)=1$.

These so-called "belief propagation" equations (8) and (9) are exact for a tree. To see this for Eq. (8), note that the graph with all the edges from i to its neighbors removed breaks into several disconnected pieces, one piece G_j for each neighbor j of i plus one piece containing only i. Define $p_{mft,j}(\{q_{k,k\in G_j}\})$ to be the Boltzmann weight for the Potts model on G_j. Then,

$$\sum_{\{q_k\},q_i=r} p_{mft}(\{q_j\}) \propto \prod_j^{i,j\ \text{n.n.}} [\sum_{\{q_{k,k\in G_j}\}} p_{mft,j}(\{q_{k,k\in G_j}\})\exp(J\delta_{r,q_j})]$$

$$\propto \prod_j^{i,j\ \text{n.n.}} [\sum_{q_j=1}^{q} \exp(J\delta_{r,q_j})p_{ji}(q_j)],$$

and similarly for Eq. (9). A more detailed exposition is in [8].

The belief propagation algorithm makes the very effective approximation of using these same equations for an arbitrary graph, even if it has loops. This is most effective if the density of loops is small and has been proven in many applications. One source of error in our approach is the appearance of loops in the network. For the randomly generated test

networks below, however, loops become sparse in the limit of large N. Further, in our tests on networks with finite N and on real-world networks the method appears to work very well in practice. A second source of error is our use of the mean field instead of belief propagation for the long-range links. This is done partly by necessity: if instead we wished to apply belief propagation to Eq. (3) without using the mean-field approximation, we would find that we needed N^2-N different variables $p_{ij}(r)$ since all nodes are connected, greatly slowing the algorithm. However, we believe that for N large, the mean-field approximation is justified since, for large N, the fluctuations in $\rho(r)$ are small compared to the average.

There are many ways to solve the belief propagation equations. We choose to initialize each of the $p_i(r) = 1/q + \epsilon_i(r)$, where $\epsilon_i(r)$ is chosen randomly and is small, and similarly for each of the $p_{ij}(r)$. We then perform a fixed number of iterations: on each iteration we first randomly select one directed edge i,j and then update the corresponding function $p_{ij}(r)$ by replacing its current value with 0.75 times its current value plus 0.25 times the value given by solving Eq. (9). We then randomly select one node i and replace the function $p_i(r)$ by 0.75 times its current value plus 0.25 times the value given by solving Eq. (8). After updating $p_i(r)$ we update $\rho(r)$ and $m_i(r)$, thus solving the belief propagation and mean-field equations simultaneously.

We have tested this algorithm on several problems. We first consider the four-groups problem. In this case, we take a randomly generated network of $N=128$ nodes, divided into four communities of 32 nodes each. After generating the network, we run the community detection algorithm, to find the most likely assignment of communities. We measure the algorithm's performance by determining the fraction of nodes whose community it identifies correctly. There is some arbitrariness in how this fraction of correctly identified nodes is defined, which is connected with the arbitrariness in the labeling of communities: one can permute the community labels as desired, given the Potts symmetry of Eq. (3). To resolve this arbitrariness, we follow the stringent definition adopted in [11] for the accuracy.

We choose p_{in} and p_{out} so that each node has on average z_{in} connections to other nodes in the same community and z_{out} connections to nodes in different communities. We pick $z_{in}+z_{out}=16$, and consider a range of values of z_{in}, the average number of intracommunity links. For large z_{in}, the community detection problem is easier, as the effect of the community structure is much more clear.

The results of this procedure are shown in Fig. 1. As seen, with 100 000 iterations, the algorithm is highly accurate. With 50 000 iterations, a slight decrease in accuracy is noticed for small z_{in}, and the accuracy goes down significantly at 10 000 iterations (by increasing to 200 000 iterations, a slight improvement is noted for $z_{in}<8$). For the first three curves, we used the choice of constants in Eq. (4). Since these constants depend on the given z_{in}, z_{out}, something which may not be known for an arbitrary network, we repeated the algorithm with a particular fixed choice of constants, $\exp(J)=6$, $\exp(NJ')=10^{-10}$. This choice of constants was chosen completely arbitrarily, but as seen the algorithm still works well, with a much higher accuracy than the Newman-Girvan algorithm. However, the Newman-Girvan algorithm has some advantages in terms of picking the most optimal number of different communities into which to divide the network, while the present algorithm takes the number of communities as an input. Indeed, the present method outperforms all others listed in [3], with the exception of a simulated annealing method [12] which offers comparable accuracy but may not be as fast.

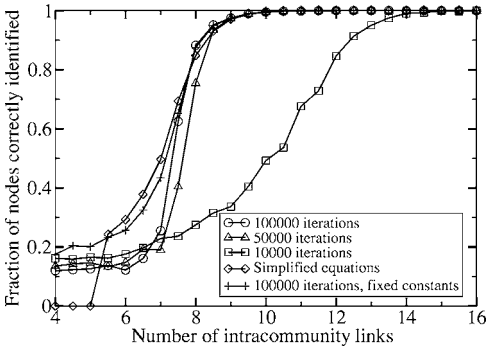

FIG. 1. Accuracy of belief propagation. Circles, squares, and triangles represent different numbers of iterations, using constants from Eq. (4). Pluses represent a different, fixed choice of constants described in text.

We can accelerate the algorithm by simplifying the belief propagation equations: for each site we track only the beliefs $p_i(q_i)$ and iterate the equations

$$p_i(r) = Z_i^{-1} \prod_{j,\langle ij\rangle} \exp[h_i(r)]\{\exp(J)p_j(r) + [1 - p_j(r)]\}. \tag{10}$$

This simplified method is most appropriate for networks with large z where it becomes faster than the belief propagation method by roughly a factor z as there are fewer equations to solve. As seen in Fig. 1, this method is almost as accurate as the belief propagation algorithm for $z_{in} \geqslant 8$, and actually performs better for $z_{in} \leqslant 7.5$. We believe that the improved performance is a result of the fact that the simplified equations more easily break symmetry. Tests on networks with a lower coordination number showed the difference more clearly, as given in Fig. 2 for a network with four groups and $z_{in}+z_{out}$

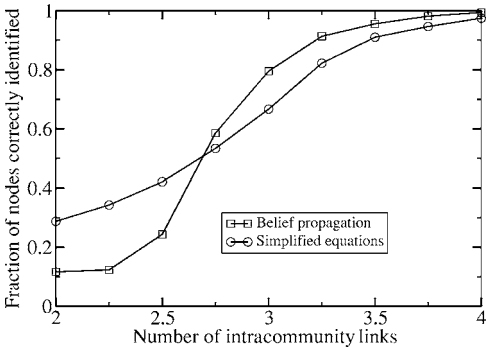

FIG. 2. Accuracy of belief propagation algorithms for $z_{in}+z_{out}=4$.

=4. Another interesting approach to speed the algorithm might be to reduce the equations to linear programming [13].

We have also tested the belief propagation algorithm on simple networks, such as dividing N nodes arranged on a straight line with connections between nearest neighbors on the line into two different communities, as well as the Zachary karate club network [14]. The relaxation of the algorithm needed to be done slightly more slowly than described above (ten randomly chosen nodes were relaxed before each edge was relaxed), and some care was taken about the constants due to the lower coordination number in order to obtain convergence: with poor choices the beliefs $p_i(r)$ oscillated randomly. After finding these constants on the straight line network, the algorithm was tested on the Zachary network, and identified the communities accurately, with one error of placing node 10 in the wrong community, as labeled in the figure in [4]. Interestingly, after convergence of the equations, for almost all nodes i, the maximum over r of $p_i(r)$ was greater than 0.99, except for node 9 where it was only 0.98 and node 10 where it was only 0.87; these two nodes have roughly equal connections to both communities.

We have expressed community detection as an inference problem, providing a formulation of the problem in statistical mechanical terms. We then applied the belief propagation method to solve the resulting statistical problem. The results are accurate; however there are a number of questions that should be addressed as well as possible extensions. First, there is some questions about picking the constants J,J'. In some cases, especially on test networks with a low coordination number, a poor choice of constants leads to either a lack of convergence of the belief propagation equations, or else convergence to a solution in which $p_i(r) \to 1/q$ so that the spontaneous symmetry breaking is absent. In both cases the algorithm performs poorly at finding the communities. The former case requires a slower relaxation of the equations, while the latter case requires an increase in the constants J,J'. Fortunately, both of these cases can be detected by looking at the $p_i(r)$ as the algorithm runs, and then corrected, so that the algorithm warns of its possible failure in these cases. Even on the simple case of N nodes on a straight line above, a poor choice of constants can lead to failure to converge. However, we emphasize that for every network we tried, it was possible to find a choice of J,J' for which the algorithms converged to a symmetry breaking solution, and for every such case this appeared to be a good solution. For randomly generated networks as above, one can obtain the correct values of J,J' from Eq. (4) and these appear to always work. It would be interesting, however, to test this algorithm on very heterogeneous networks.

The next question is the scaling of the algorithm with system size. Accurate results were found with 50 000 iterations, or $50\,000/(128\times 16)\approx 24$ iterations per edge. For a general network, we expect that if there are few links between communities, then the number of iterations required per edge will be proportional to the phase-ordering time for a given community under the appropriate dynamics. This phase-ordering time typically scales [15] as some power of the relevant length scale for a community, and for many networks this length scale is of order $\ln(N)$. We thus expect for a network with average coordination number z that the time will typically be of order $zN\ln(N)^{\alpha}$ for some α.

I thank M. Chertkov and E. Ben-Naim for useful discussions. This work was supported by U.S. DOE Grant No. W-7405-ENG-36.

[1] M. E. J. Newman, Eur. Phys. J. B **38**, 321 (2004).
[2] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas, Phys. Rev. E **68**, 065103(R) (2003); P. Holme, M. Huss, and H. Jeong, Bioinformatics **19**, 532 (2003).
[3] For a variety of methods, see references in L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, J. Stat. Mech.: Theory Exp. 2005, P09008.
[4] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
[5] J. Reichardt and S. Bornholdt, Phys. Rev. Lett. **93**, 218701 (2004); Phys. Rev. E **74**, 016110 (2006).
[6] M. Blatt, S. Wiseman, and E. Domany, Phys. Rev. Lett. **76**, 3251 (1996); C. M. Fortuin and P. W. Kasteleyn, Physica (Amsterdam) **57**, 536 (1972); M. Bengtsson and P. Roivainen, Int. J. Neural Syst. **6**, 119 (1995).
[7] R. G. Gallager, *Low Density Parity Check Codes* (MIT Press, Cambridge, MA, 1963).
[8] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms* (Cambridge University Press, Cambridge, U.K., 2003).
[9] N. Sourlas, Nature (London) **339**, 29 (1989).
[10] H. A. Bethe, Proc. R. Soc. London, Ser. A **150**, 552 (1935); R. Peierls, Proc. Cambridge Philos. Soc. **32**, 477 (1936).
[11] M. E. J. Newman, Phys. Rev. E **69**, 066133 (2004).
[12] R. Guimerà, M. Sales-Pardo, and L. A. N. Amaral, Phys. Rev. E **70**, 025101(R) (2004).
[13] M. Chertkov and M. G. Stepanov, e-print cs.IT/0601113.
[14] W. W. Zachary, J. Anthropol. Res. **33**, 452 (1977).
[15] A. J. Bray, Adv. Phys. **43**, 357 (1994).