Efficient discovery of overlapping communities in massive networks

Prem K. Gopalan1 and David M. Blei

Department of Computer Science, Princeton University, Princeton, NJ 08540

Edited by Chris H. Wiggins, Columbia University, New York, NY, and accepted by the Editorial Board July 1, 2013 (received for review December 18, 2012)

Detecting overlapping communities is essential to analyzing and exploring natural networks such as social networks, biological networks, and citation networks. However, most existing approaches do not scale to the size of networks that we regularly observe in the real world. In this paper, we develop a scalable approach to community detection that discovers overlapping communities in massive real-world networks. Our approach is based on a Bayesian model of networks that allows nodes to participate in multiple communities, and a corresponding algorithm that naturally interleaves subsampling from the network and updating an estimate of its communities. We demonstrate how we can discover the hidden community structure of several real-world networks, including 3.7 million US patents, 575,000 physics articles from the arXiv preprint server, and 875,000 connected Web pages from the Internet. Furthermore, we demonstrate on large simulated networks that our algorithm accurately discovers the true community structure. This paper opens the door to using sophisticated statistical models to analyze massive networks.

network analysis | Bayesian statistics | massive data

Community detection algorithms (1–17) analyze networks to find groups of densely connected nodes. These algorithms have become vital to data-driven methods for understanding and exploring network data such as social networks (4), citation networks (18), communication networks (19), and networks induced by scientific observation [e.g., gene regulation networks (20)].

Community detection is important for both exploring a network and predicting connections that are not yet observed. For example, by finding the communities in a large citation graph of scientific articles, we can make hypotheses about the fields and subfields that they contain. By finding communities in a large social network, we can more easily make predictions to individual members about who they might be friends with but are not yet connected to.

In this paper, we develop an algorithm that discovers communities in modern real-world networks. The challenge is that real-world networks are massive—they can contain hundreds of thousands or even millions of nodes. We will examine a network of scientific articles that contains 575,000 articles, a network of connected Web pages that contains 875,000 pages, and a network of US patents that contains 3,700,000 patents. Most approaches to community detection cannot handle data at this scale.

There are two fundamental difficulties to detecting communities in such networks. The first is that many existing community detection algorithms assume that each node belongs to a single community (1, 3–7, 14–16). In real-world networks, each node will likely belong to multiple communities and its connections will reflect these multiple memberships (2, 8–13, 17). For example, in a large social network, a member may be connected to coworkers, friends from school, and neighbors. We need algorithms that discover overlapping communities to capture the heterogeneity of each node’s connections.

The second difficulty is that existing algorithms are too slow. Many community detection algorithms iterate by analyzing each pair of nodes, regardless of whether the nodes in the pair are connected in the network (5, 6, 10). Consequently, these algorithms run in time squared in the number of nodes, which makes analyzing massive networks computationally intractable. Other algorithms avoid computation about unconnected nodes (2–4, 7–9, 11–17). These methods are more efficient, but either make too simple assumptions, are still difficult to scale, or have difficulty with prediction.

Our algorithm addresses these difficulties. It discovers the hidden overlapping communities in massive networks, and its results can be used to explore, understand, and form predictions about their structure. Fig. 1 gives an example. This is a subgraph of a network of 575,000 scientific articles on the arXiv preprint server (21); each link denotes that an article cites or is cited by another article. Our algorithm analyzes this network, discovering overlapping communities among the citations. It assigned multiple communities to each article and a single community to each link. Many articles mostly link to other articles within their main community. However, the article “An alternative to compactification” (22) is different—it links to multiple communities, which suggests that it relates to multiple fields. Identifying nodes in large networks that bridge multiple communities is one way that our algorithm gives insights into the structure of the network.

Our algorithm identifies hundreds of overlapping communities among millions of nodes in a matter of hours. It is fast because of its simple structure: (1) subsample a subgraph from the full graph; (2) analyze the subgraph under the algorithm’s current estimate of the communities; (3) update this estimate of the communities, based on the analysis from the previous step; (4) repeat.

This powerful algorithmic structure is efficient because it only analyzes a subgraph of the network at each iteration. These subgraphs can be as large or as small as is computationally feasible, and can be designed to maximize the statistical information for efficiently finding communities. Furthermore, the algorithm does not require that the network be fully observed before beginning to estimate communities; its algorithmic structure naturally interleaves data collection with data analysis.

What we will show below is that our algorithm emerges when we take a Bayesian approach to detecting overlapping communities. In particular, we posit a probabilistic model of networks (23) where each node can belong to multiple communities (10). We then analyze a network by computing the posterior, the conditional distribution of the hidden communities given the observed network. The efficient structure of the algorithm—iteratively subsampling the network and updating an estimate of the hidden communities—emerges when we approximate this conditional distribution with variational methods (24) in combination with stochastic optimization (25, 26).

In the rest of the paper, we describe a model of overlapping communities (10) and present our efficient algorithm for computing with it. We demonstrate the capabilities of this analysis on three large real-world networks and report on a study of large simulated networks where the community structure is known.

Author contributions: P.K.G. and D.M.B. designed research, performed research, contributed new analytic tools, analyzed data; and wrote the paper.

The authors declare no conflict of interest.

This article is a PNAS Direct Submission. C.H.W. is a guest editor invited by the Editorial Board.

Freely available online through the PNAS open access option.

1To whom correspondence should be addressed. E-mail: pgopalan@cs.princeton.edu.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1221839110/-/DCSupplemental.
One of the main advantages of taking a probabilistic approach to network analysis is that the models and algorithms are reusable in more complex settings. Our strategy for analyzing networks easily extends to other probabilistic models, such as those taking into account degree distribution or node attributes beyond the network. The approach we develop here opens the door to using sophisticated statistical models to analyze massive networks.

The Model and Algorithm

We describe a Bayesian model of overlapping communities and our scalable algorithm for computing with it.

A Mixed-Membership Stochastic Blockmodel. We describe the model by its probabilistic generative process of a network. In this process, the community memberships will be encoded as hidden random variables. Given an observed network, such as a social network of friendship ties, we discover the hidden community structure by estimating its conditional distribution.

Classical community membership models, like the stochastic blockmodel (5, 6, 27), assume that each node belongs to just one community. Such models cannot capture that a particular node belongs to multiple communities. This is taken from an analysis of the full 575,000 node network.

The model assumes there are \( K \) communities and that each node \( i \) is associated with a vector of community memberships \( \theta_i \). This vector is a distribution over the communities—it is positive and sums to 1. For example, consider a social network and a member for whom one-half of her friends are from work and the other half are from her neighborhood. For this node, \( \theta \) would place one-half of its mass on the work community and the other half on the neighborhood community.

To generate a network, the model considers each pair of nodes. For each pair \( (i,j) \), it chooses a community indicator \( z_{ij} \) from the \( i \)th node’s community memberships \( \theta_i \) and then chooses a community indicator \( z_{ji} \) from \( \theta_j \). If these indicators point to the same community, then it connects nodes \( i \) and \( j \) with high probability; otherwise, they are likely to be unconnected.

These assumptions capture that the connections between nodes can be explained by their memberships in multiple communities, even if we do not know where those communities lie. To see this, we consider a single pair of nodes \( (i,j) \) and compute the probability that the model connects them, conditional on their community memberships. This computation requires that we marginalize out the value of the latent indicators \( z_{ij} \) and \( z_{ji} \).

Let \( \beta_k \) be the probability that two nodes are connected given that their community indicators are both equal to \( k \). For now, assume that if the indicators point to different communities then the two nodes have zero probability of being connected. (In the full model, they will also have a small probability of being connected when the indicators are different, but this simplified version gives the intuition.) The conditional probability of a connection is as follows:

\[
p(y_{ij} = 1|\theta_i, \theta_j) = \beta_\text{comm} \phi_k \beta_k. \tag{1}
\]

The first two terms represent the probability that both nodes draw an indicator for the \( k \)th community from their memberships; the last term represents the conditional probability that they are connected given that they both drew that indicator. (The parameter \( \beta_k \) relates to how densely connected the \( k \)th community is.) The probability that nodes \( i \) and \( j \) are connected will be high when \( \theta_i \) and \( \theta_j \) share high weight for at least one community, such as if the social network members attended the same school; it will be low if there is little overlap in their communities. The summation marginalizes out the communities, capturing that the model is indifferent to which communities the nodes have in common. The model captures assortativity—nodes with similar memberships will more likely connect to each other (28, 29).

We described the probability that governs a single connection between a pair of nodes. For the full network, the model assumes the following generative process:

1. For each node, draw community memberships \( \theta_i \sim \text{Dirichlet}(\alpha) \).
2. For each pair of nodes \( i \) and \( j \), where \( i < j \):
   (a) Draw community indicator \( z_{ij} \sim \theta_i \).
   (b) Draw community indicator \( z_{ji} \sim \theta_j \).
   (c) Draw the connection between them from
   \[
p(y_{ij} = 1|z_{ij}, z_{ji}) = \begin{cases} 
   \beta_{z_{ij}} & \text{if } z_{ij} = z_{ji} \\
   \epsilon & \text{if } z_{ij} \neq z_{ji}.
   \end{cases}
   \]

This defines a joint probability distribution over the \( N \) per-node community memberships \( \Theta \), the per-pair community indicators \( z \), and the per-node community membership vectors \( \theta \).
and the observed network \( y \) (both links and nonlinks). We will use this as a model of an undirected graph, but it easily generalizes to the directed case.

Given an observed network, the model defines a posterior distribution—the conditional distribution of the hidden community structure—that gives a decomposition of the nodes into \( K \) overlapping communities. In particular, the posterior will place higher probability on configurations of the community memberships that describe densely connected communities. With this posterior, we can investigate the set of communities each node participates in and which specific communities are responsible for each of the observed links. In this sense, our algorithm discovers link communities (2, 9). Our visualization in Fig. 1 illustrates this posterior superimposed on a subgraph of the original network.

As for many interesting Bayesian models, however, this posterior is intractable to compute. Furthermore, existing approximation methods like Markov chain Monte Carlo (30) or variational inference (24) are inefficient for real-world-sized networks because they must iteratively consider all pairs of nodes (5, 6, 10). In the next section, we develop an efficient algorithm for approximating the posterior with massive networks.

Finally, we note an exception. The Poisson community model (2) is a probabilistic model of overlapping communities that avoids the all-pairs computation and can be efficiently estimated. We show below that the Poisson model can be good for uncovering true community structure. But we also found that it cannot compute held-out probabilities of links, which makes it ineffective for prediction or for model metrics based on prediction (e.g., to select \( K \)).

**Posterior Inference.** Our modeling assumptions capture the intuition that each node belongs to multiple communities. To examine observed networks under these assumptions, we compute the posterior distribution of the community structure,

\[
p(\theta, z | y) = \frac{p(\theta, z, y)}{p(y)}. \tag{2}
\]

This posterior cannot be computed exactly. (In practice, the community densities \( p_i \) are also hidden variables. For simplicity we treat them here as fixed parameters but give all details in SI Text.)

The numerator is easy to compute. It is the joint distribution defined by the modeling assumptions. The problem is with the denominator. It is the marginal probability of the data, which implicitly sums over all possible hidden community structures,

\[
p(y) = \int_{\theta} \sum_z p(\theta, z, y). \tag{3}
\]

Computing the marginal requires a complicated integral over \( N \) simplicial variables and a summation over the \( K^N \) configurations of community indicators. This is exacerbated by the number of nodes \( N \) being large. Thus, we approximate the posterior.

As we described above, our algorithm to approximate the posterior iterates between subsampling the network, analyzing the subsample, and updating the estimated community structure. This computational structure lets us approximate the posterior with massive networks. It emerges when we adapt two key ideas to the problem: mean-field variational inference and stochastic optimization.

**Mean-Field Variational Inference.** Variational inference is a powerful approach to approximate posterior inference in complex probabilistic models (24). It has been adapted to a variety of probabilistic models, although its roots are in the statistical physics literature (31). Variational inference algorithms approximate the posterior in Eq. 2 by defining a parameterized family of distributions over the hidden variables and then fitting the parameters to find a distribution that is close to the posterior.

Closeness is measured with Kullback–Leibler (KL) divergence (32). Thus, the problem of posterior inference becomes an optimization problem.

The mean-field variational family independently considers each hidden variable with a different parameterized distribution. In our model, the mean-field variational family is as follows:

\[
q(\theta, z) = \prod_{n=1}^{N} q(\theta_n | y_n) \prod_{r<s} q(z_{r<s} | \phi_{r<s}) q(z_{r<s} | \phi_{r<s}). \tag{4}
\]

Each factor is in the same family as the corresponding component in the model, but there is a different independent distribution for each instance of each hidden variable.

For example, the model contains a single Dirichlet distribution that specifies the prior over community memberships \( \theta_i \). However, the variational distribution \( q(\theta | y) \) has a different Dirichlet distribution for each node. Once fit, the variational parameter \( \gamma_i \) captures the posterior distribution of the \( n \)th node’s community memberships. Similarly, the model defines the distribution of community indicators based on the corresponding nodes’ community memberships. However, the variational distributions for those indicators are freely parameterized discrete distributions. Once fit, the parameters \( \phi_{r<s} \) and \( \phi_{r<s} \) describe discrete distributions that capture the posterior distribution of the indicators when considering the possible connection between \( i \) and \( j \). Each variable having its own distribution makes the variational family very flexible. With \( N \) nodes, it can uniquely describe each node’s community memberships and which community is activated for each pair of nodes’ possible connection.

The variational family \( q(\theta, z) \) reflects the hidden structure of the observed network when we optimize the variational parameters to minimize the KL divergence to the posterior. Thus, the variational problem is to solve the following:

\[
q^*(\theta, z) = \arg \min_{q} \text{KL}(q(\theta, z) || p(\theta, z | y)). \tag{5}
\]

We then use the optimal \( q^* \) as a proxy for the true posterior, for example to identify communities in the data or to make predictions about as-yet-unseen links. The optimization connects the variational distribution to the data.

Unfortunately, we cannot minimize the KL divergence directly—it is difficult to compute for the same reason that the posterior is difficult to compute—but we can optimize an objective function that is equal to the negative KL divergence up to a constant. (Its optimum is the same as the optimum of the KL objective.) Let \( H[\cdot] \) be the entropy of a distribution. The variational objective is

\[
L(\gamma, \phi) = \mathbb{E}[-\log p(\theta, z, y)] + H[q(\theta, z)],
\]

where the expectation is taken with respect to the variational distribution. This objective is a function of the variational parameters in Eq. 4. It relates to the free energy in statistical physics; the first term is the internal energy, with the temperature set to 1.

Maximizing this objective is equivalent to minimizing the KL divergence to the posterior. Intuitively, the first term captures how well \( q(\theta, z) \) describes a distribution that is likely under the model, keeping both the priors and data in mind through the joint distribution; the second term encourages the variational distribution to be entropic, i.e., this protects it from “overfitting.” Traditional variational inference optimizes the objective with coordinate ascent, iteratively optimizing each variational parameter while holding the others fixed. This has been a successful approach for probability models of small networks (6, 10).

However, traditional variational methods for overlapping community detection do not scale well to real-world-sized networks; previous work has only analyzed networks in the hundreds of nodes. The problem is that there are \( O(N^2) \) terms in the objective
function and $O(N^2)$ variational parameters. Coordinate ascent inference must consider each pair of nodes at each iteration (10), but even a single pass through a large network can be prohibitive. Our variational inference algorithm avoids this issue through stochastic optimization.

**Stochastic Optimization of the Variational Objective.** Stochastic optimization algorithms follow noisy estimates of the gradient of an objective with a decreasing step size. In their classic paper, Robbins and Monro showed that with certain step-size schedules, such algorithms provably lead to the optimum of a convex function (25). (In our case, they provably lead to a local optimum.) Since the 1950s, stochastic optimization has blossomed into a field of its own (33).

Stochastic optimization is particularly efficient when the objective is a sum of terms, as is the case for the variational objective of our model. In these settings, we cheaply compute a stochastic gradient by first subsampling a subset of terms and then forming an appropriately scaled gradient. The scaled gradient is a random variable whose expectation is the true gradient.

Specifically, the variational objective for our model contains a sum of terms for each pair of nodes. Thus, our algorithm iteratively subsamples a subset of pairs and then updates its current estimate of the community structure by following a scaled gradient computed only on that subset. (By “community structure,” we mean the $N \times K$ parameters $\gamma$, which describe the posterior distribution of each node’s community memberships.) This is a form of stochastic variational inference algorithm (26). At iteration $t$, we

1. Subsample a set of pairs of nodes $\mathcal{S}$.
2. For each pair $(i, j) \in \mathcal{S}$, use the current community structure to compute the indicator parameters $\phi_{i \rightarrow j}$ and $\phi_{j \rightarrow i}$.
3. Adjust the community memberships $\gamma$.
4. Repeat.

The details of how we find the optimal indicator parameters and how we adjust the community memberships are in *SI Text*. What is important about our algorithm is that it does not require analyzing all $N^2$ pairs at each iteration and that it is a valid stochastic optimization algorithm of the variational objective. It scales to massive networks.

Our algorithm is flexible in terms of how we sample the subset of pairs in step 1. We can analyze all of the pairs associated with a sampled node; or we can use a subsampling technique that makes data collection easier, for example if the network is stored in a distributed way. We have explored several methods of subsampling pairs of nodes:

- Sample uniformly from the set of all pairs.
- Sample a node and select its linked pairs.
- Sample a node and select all its pairs (links and nonlinks).

Naively applied, biased sampling strategies lead to biases in approximate posterior inference. In *SI Text*, we show how to correct for these biases. Using these strategies can lead to faster convergence of the variational distribution (34).

We emphasize that our algorithm does not prune the network to make computation manageable (18). Rather, it repeatedly subsamples subgraphs at each iteration. Furthermore, we do not need to have collected the entire network to run the algorithm. Because it operates on subgraphs, it gives a natural approach for interleaving data collection and model estimation.

**The Number of Communities.** Probabilistic models of community detection require setting the number of communities, and in typical applications we will want to set this number based on the data. In our empirical study, we addressed this model selection problem in two ways. One was by evaluating the predictive performance of the model for varying numbers of communities. A second way was to set the number of communities as part of the initialization procedure of the variational distribution. This is detailed in *SI Text*. It works well and is faster than the predictive approach.

**A Study of Real and Synthetic Networks**

We studied our algorithm on real and synthetic networks. With real networks, we demonstrate how it can help us explore massive data: on networks with millions of nodes, it identifies overlapping communities and the nodes that bridge them. On synthetic data, where the ground truth is known, we confirm that it accurately identifies the overlapping communities.

**Exploring Real-World Network Data.** We first show how our algorithm can be used to study massive real-world networks. We analyzed two citation networks: a network of 575,000 articles from the arXiv preprint server (21) and a network of 3,700,000 patents from the US patent network (35). In these networks, a link indicates that one document cites another. We also analyzed a large network of 875,000 Web pages from Google (36). (These data did not contain the descriptions of the nodes that are required to visualize the communities. Our quantitative analyses of this network are in *SI Text*.) In all networks, we treated the directed links as undirected—the presence of a link is evidence of similarity between the nodes and is independent of direction. [This is common in hyperlink graph analysis (1).] These networks are much larger than what can easily be analyzed with previous approaches to computing with mixed-membership stochastic blockmodels (10). Although we note that several efficient methods have recently been developed for blockmodels without overlapping communities (14–16), we analyze a network by setting the number of communities $K$ and running the stochastic inference algorithm. (Our software is available at [https://github.com/premgopalan/svinet](https://github.com/premgopalan/svinet). More details about these fits are in *SI Text*.) This results in posterior estimates of the community memberships for each node and posterior estimates of the community assignments for each node pair (i.e., for each pair of nodes, estimates of which communities governed whether they are connected). With these estimates, we visualize the network according to the discovered communities.

**Scientific Articles from arXiv.** The arXiv network (21) contains scientific articles and citations between them. Our large subset of the arXiv contains 575,000 physics papers. We ran stochastic inference to discover 200 communities.

Fig. 1 illustrates a subgraph of the arXiv network and demonstrates the structure that our algorithm uncovered. In the model, each node $i$ contains community memberships $\theta_i$ and each link $(i, j)$ is assigned to one of the $K$ communities. In the figure, we colored each link according to the peak of the approximate posterior $p(\gamma_{i,j} | \gamma_{i,j}, \gamma_{j,i})$. This suggests within which communities and to what degree each paper has had an impact. (We note that most of the links attached to highly cited articles are incoming links, so visualizing these links reveals the communities influenced by the paper.)

The central article in Fig. 1 is the highly cited article “An alternative to compactification” (22), which was published in 1999. The article proposes a simple explanation to one of the most important problems in physics: Why is the weak force 10^32 times stronger than gravity? The paper’s external tag (given by the authors) suggests it is primarily a theoretical paper. It has had, however, an impact on a diverse array of problems including certain astrophysics puzzles regarding the structure of the universe (37) and the confrontation between general relativity and experiment (38).

In analyzing the full network of citations, our algorithm has captured how this article has played a role in multiple subfields. It assigned it to membership in nine communities and gave it a high posterior bridgeness score (39), a measure of how strongly it bridges multiple communities. We note that bridgeness is a function of known community memberships. In our networks, the communities are not observed. Thus, we estimated the posterior using our algorithm and then computed the expected bridgeness.
In the subgraph of Fig. 1, the link colors correspond to the research communities associated with the links. We visualize the four top communities that link to this article: “High Energy Physics: Theory,” “High Energy Physics: Phenomenology,” “General Relativity and Quantum Cosmology,” and “Astrophysics.” (Naming and interpreting communities is a difficult problem in unsupervised community detection. For visual convenience, we examine the external tags given to the articles and name each community by its most common tag. Note the algorithm does not have access to the tags.) We emphasize that the citations alone cannot reveal the role of an article in its citation graph—we executed this analysis by first discovering the communities with our algorithm and then using those discovered communities to compute quantities, like bridgeness (39) and link color, that require community assignments.

As an example of a different kind of article, consider “Cosmological constant—the weight of the vacuum” (40). This article has 1,117 citations in the dataset, on the same order as ref. 22. It discusses the theoretical and cosmological aspects of the cosmological constant. Our algorithm finds that this article has a lower bridgeness, and membership in only two communities. Both communities are dominated by the “Astrophysics” subject tag, with the other significant tag being “General Relativity and Quantum Cosmology.” Detecting these two kinds of articles highlights an advantage of this type of analysis. By discovering the hidden community structure, we can separate articles (of similar citation count) that have had interdisciplinary impact from those with impact within their particular fields.

We have illustrated a small subgraph of this large network, centered around a specific article. Across the whole network, we can use the posterior bridgeness to filter and find a collection of articles that have had interdisciplinary impact. In SI Text we show the top 10 papers in the arXiv network by posterior bridgeness. The top scientific articles in the arXiv network have a wide impact, as they concern data, parameters, or theory applied in various subfields of physics. For example, the top article, “Maps of dust infrared emission for use in estimation of reddening and cosmic microwave background radiation foregrounds,” (41) constructs an accurate full sky map of the dust temperature useful in the estimation of cosmic microwave background radiation. This filtering demonstrates the practical potential for unsupervised analysis of large networks. The posterior bridgeness score, a function of the discovered communities, helps us focus on a class of nodes that is otherwise difficult to find.

**US Patents.** The National Bureau of Economic Research maintains a large dataset of US patents (35). It contains 3,700,000 patents granted between 1975 and 1999 and the citations between them. We analyzed this network, setting the number of communities to 1,000.

Fig. 2 illustrates a subgraph of the patents data that reveals overlapping community structure around “Process for producing porous products” (42). This patent was issued in 1976 and describes an efficient process for producing highly porous materials from tetrafluoroethylene polymers. It has influenced the design of many everyday materials, such as waterproof laminate, adhesives, printed circuit boards, insulated conductors, dental floss, and strings of musical instruments. Our algorithm assigned it a high posterior bridgeness and membership in 39 communities. The classification tags of the citing patents confirm that it has influenced several areas of patents: Synthetic Resins or Natural Rubbers, Prosthesis, Stock Material, Plastic and Nonmetallic Article Shaping, Adhesive bonding, Conductors and Insulators, and Web or Sheet. Fig. 2 illustrates the top communities for this patent, found by our algorithm.

We also studied a patent with a comparable number of citations but with significantly lower bridgeness. “Self-controlled release device for administering beneficial agent to recipient” (43) concerns a novel osmotic dispenser for continually administering agents, e.g., ophthalmic drugs. It has 339 citations, comparable to the 441 of ref. 42, but a much lower bridgeness score. Our algorithm assigned it to seven communities, with the classification tags mostly restricted to “Drug: Bio-Affecting and Body Treating Compositions” and “Surgery.”

**Comparisons to Ground Truth on Synthetic Networks.** We demonstrated that our algorithm can help explore massive real-world networks. As further validation, we performed a benchmark comparison on synthetic networks where the overlapping communities are known. We used the “benchmark” tool (44) to synthesize networks with the number of nodes ranging from one thousand to one million.

We compared our algorithm to the best existing algorithms for detecting overlapping communities (2, 8, 9, 11–13, 17). Each algorithm analyzes the (unlabeled) network and returns both the...
The performance of scalable algorithms on synthetic networks with overlapping communities. The numbers of nodes in each network span ten thousand to one million, and for each network size we generated five networks. Our stochastic inference algorithm (SVI) outperforms scalable alternatives, the INFOMAP algorithm (INF) (13) and the COPRA algorithm (COP) (12), while performing as well as the Poisson community model (POI) (2). We measure accuracy with normalized mutual information (NMI) (44).

We also compared with many other methods that could not scale up to one million nodes; see SI Text for a full table of results.

Discussion

We have developed and studied a scalable algorithm for discovering overlapping communities in massive networks. Our approach naturally interlaces subsampling the network and reestimating its community structure. We focused on a specific Bayesian model but we emphasize that this strategy can be used to accommodate many kinds of assumptions. For example, we can posit varying degree distributions to better capture the expected properties of real networks or use Bayesian nonparametric assumptions (45) to infer the number of communities within the analysis. In general, with the ideas presented here, we can use sophisticated statistical models to analyze massive real-world networks.

ACKNOWLEDGMENTS.

D.M.B. is supported by Office of Naval Research Grant N00014-11-1-0651, National Science Foundation CAREER 0745520, BigDATA IIS-1247664 and The Alfred P. Sloan Foundation.
Supporting Information

Gopalan and Blei 10.1073/pnas.1221839110

SI Text

Introduction

This document is organized as follows. We first develop the stochastic variational inference (SVI) algorithm (1) for the mixed-membership stochastic block model (MMSB) (2) described in the article. Each iteration of the algorithm subsamples the network and updates its estimate of the community structure. We extend the algorithm to allow for nonuniform sampling from the network, and study a number of sampling strategies. We then develop SVI with link sampling, an algorithm whose per-iteration complexity scales linearly in the number of links. Finally, we present supporting material for the empirical study on real and synthetic networks.

SVI

The article describes a subclass of the MMSB (2, 3) that is appropriate for community detection in assortative undirected networks. In this section, we present SVI for the MMSB (3).

In variational inference, we define a family of distributions over the hidden variables \( q(\theta, \beta, z) \) and find the member of that family that is closest to the true posterior. Closeness is measured with the Kullback–Leibler (KL) divergence (4). We use the mean-field family, under which each variable is endowed with its own distribution and its own variational parameter. This allows us to tractably optimize the parameters to find a local minimum of the KL divergence. The mean-field variational family for the MMSB, with \( N \) nodes and \( K \) communities and with Beta priors placed over the community strengths \( \beta \), is as follows:

\[
q(\beta, \theta, z) = \prod_{k=1}^{K} q(\beta_k | \lambda_k) \prod_{n=1}^{N} q(\theta_n | \gamma_n) \prod_{i<j} q(z_{ij} | \phi_{ij}). \tag{S1}
\]

Here, the variational distributions are \( q(z_{ij} = k) = \phi_{ij} | k, q(\theta_n) = \text{Dirichlet}(\gamma_n), \) and \( q(\beta_k) = \text{Beta}(\lambda_k) \). The posterior over \( z \) is parameterized by the interaction parameters \( \phi \), the posterior over \( \theta \) is parameterized by the community memberships \( \gamma \), and the posterior over \( \beta \) is parameterized by the community strengths \( \lambda \).

Minimizing the KL divergence between \( q \) and the true posterior is equivalent to optimizing an “evidence lower bound” (ELBO) \( \mathcal{L} \), a bound on the log likelihood of the observations (5, 6). The ELBO is as follows:

\[
\mathcal{L} = \sum_k \mathbb{E}_\gamma \left[ \log p(\beta_k \mid \eta) \right] - \sum_k \mathbb{E}_\gamma \left[ \log q(\beta_k \mid \lambda_k) \right] \\
\quad + \sum_n \mathbb{E}_\gamma \left[ \log p(\theta_n \mid \alpha) \right] - \sum_n \mathbb{E}_\gamma \left[ \log q(\theta_n \mid \gamma_n) \right] \\
\quad + \sum_{a,b} \mathbb{E}_\gamma \left[ \log p(z_{ab} \mid \phi_{ab}) \right] + \mathbb{E}_\gamma \left[ \log p(z_{ab} \mid \phi_{ab}) \right] \\
\quad - \sum_{a,b} \mathbb{E}_\gamma \left[ \log q(z_{ab} \mid \phi_{ab}) \right] - \mathbb{E}_\gamma \left[ \log q(z_{ab} \mid \phi_{ab}) \right] \\
\quad + \sum_{a,b} \mathbb{E}_\gamma \left[ \log p(y_{ab} \mid \gamma_n, z_{ab} \mid \phi_{ab}, \beta) \right]. \tag{S2}
\]

The expectations in Eq. S2 are taken with respect to the variational distribution \( q \). Notice the first two lines in Eq. S2 contain summations over communities and nodes; we call these “global terms.” They relate to the “global variables,” which are the community strengths and community memberships. The remaining lines contain summations over all node pairs, which we call “local terms.” They depend on both the global and “local variables,” the latter being the interaction memberships.

SVI optimizes the ELBO using stochastic gradient ascent. Stochastic gradient algorithms follow noisy estimates of the gradient with a decreasing step size. These algorithms are guaranteed to converge to a local optimum if the expectation of the noisy gradient is equal to the gradient and if the step-size decreases according to a certain schedule (7). In SVI, we form noisy gradients by subsampling the network. This leads to a scalable algorithm because it avoids the expensive all-pairs sums in the ELBO.

Existing SVI methods require the data be sampled uniformly to form noisy gradients (1). We now develop an SVI algorithm that allows for nonuniform samples of links and nonlinks at each iteration. We then present SVI with link sampling, an algorithm that samples only the links of a network.

SVI with Nonuniform Sampling. SVI iteratively updates the local and global parameters. At each iteration, it first subsamples the network. It then computes the optimal local parameters of the subset—the \( (\phi_{ij}, \gamma_i, \gamma_j) \) for each sampled node pair \( (i, j) \)—given the current settings of the global parameters \( \gamma \) and \( \lambda \). Finally, it updates the global parameters using a noisy natural gradient (8) computed from the subsampled data and the optimized local parameters. The first phase is the local step; the second phase is the global step (1).

The pseudocode of SVI for the MMSB is as follows:

1. Initialize global parameters \( \gamma = (\gamma_i)_{i=1}^N, \lambda = (\lambda_k)_{k=1}^K \).
2. Subsample a set \( S \) of node pairs.
3. Local step. For each pair \( (i, j) \in S \), compute the optimal interaction parameters \( \phi_{ij} \) and \( \phi_{ji} \) as a function of the global parameters.
   - For each node \( a \), compute the community membership natural gradients \( \partial \gamma_a / \gamma_a \) and update \( \gamma_a \).
   - For each community \( k \), compute the community strength natural gradients \( \partial \lambda_k^2 / \lambda_k \) and update \( \lambda_k \).
5. Repeat.

The subsampling of the network in each iteration provides a way to plug in a variety of network sampling algorithms into the estimation procedure. However, to maintain a correct stochastic optimization algorithm of the variational objective, the subsampling method must be valid. That is, the noisy gradients estimated from the subsample must be unbiased estimates of the true gradients.

The Global Step. The global step updates the global community strengths \( \lambda \) and community memberships \( \gamma \) with a stochastic gradient of the ELBO in Eq. S2. The ELBO contains summations over all \( O(N^2) \) node pairs. Consider drawing a node pair \( (a, b) \) at random from a distribution \( g(a, b) \) over the \( M = N(N-1)/2 \) node pairs. We can rewrite the ELBO as a random function of the variational parameters that includes the global terms and the local terms associated only with \( (a, b) \). The expectation of this random function is equal in objective to Eq. S2.

For example, the term in the fifth line in Eq. S2 is rewritten as follows:
\[ \sum_{a,b} E_q [\log p(y_{ab} | z_{a-b}, z_{a-b}, \beta)] = E_q \left[ \frac{1}{g(a, b)} E_q [\log p(y_{ab} | z_{a-b}, z_{a-b}, \beta)] \right]. \] \[ \text{[S3]} \]

Evaluating the rewritten Eq. S2 for a node pair sampled from \( g \) gives a noisy but unbiased estimate of the ELBO. Following (1), the stochastic natural gradients computed from a sample pair \((a, b)\), at iteration \( t \), are as follows:

\[ \frac{\partial g'_t}{\partial \beta_k} = \alpha_k + \frac{1}{g(a, b)} \phi_{a-b,k} - \gamma_{a,k}^{-1}, \] \[ \text{[S4]} \]

\[ \frac{\partial g'_t}{\partial \phi_{a-b,k}} = \eta_k + \frac{1}{g(a, b)} \phi_{a-b,k} \phi_{a-b,k} y_{ab,i} - \lambda_{a,k}^{-1}. \] \[ \text{[S5]} \]

where \( y_{ab,0} = y_{ab} \), and \( y_{ab,1} = 1 - y_{ab} \). In practice, we sample a “mini-batch” \( S \) of pairs per update, to reduce noise.

The update in Eq. S4 can be interpreted as follows. When a single pair \((a, b)\) is sampled, we find a noisy natural gradient in Eq. S4 by computing the community memberships \( \gamma \) that would be optimal (given interaction parameters \( \phi \)) if our entire network were a multigraph containing the interaction \( y_{ab} \) repeated \( 1/g(a, b) \) times.

Once the noisy gradients are computed, the global steps follow it with an appropriate step size,

\[ \gamma - \gamma + \rho_i \frac{\partial g_i}{\partial \beta_i}; \lambda = \lambda + \rho_i \frac{\partial g_i}{\partial \lambda_i}. \] \[ \text{[S6]} \]

We require that \( \sum s^2 < \infty \) and \( \sum s = \infty \) for convergence to a local optimum (7). We set \( \rho_i(t+1)^{-k} \) for each community \( \kappa \in \{0.5, 1\} \) is the forgetting rate and the delay \( \tau_i \geq 0 \) downweights iterations (1).

**Set-based Sampling.** Our algorithm has assumed that the subset of node pairs \( S \) is sampled independently. We can relax this assumption by defining a distribution over predefined sets of pairs. These sets can be defined using information about the pairs, such as network topology, which lets us take advantage of more sophisticated sampling strategies. For example, we can define a set for each node that contains the node’s adjacent links and nonlinks. At each iteration, we sample one of these sets at random.

We set two constraints to ensure that set-based sampling results in unbiased estimates. First, the union of the sets \( S \) must be the total set of all node pairs, \( U : U = U_S \). Second, every pair \((a, b)\) must occur in some constant number of sets \( c \geq 1 \). With these conditions satisfied, we can again rewrite Eq. S2 as the sum over its global terms and an expectation over the local terms. Let \( h(i) \) be a distribution over the sets. For example, the term in the fifth line in Eq. S2 can be written as follows:

\[ \sum_{a,b} E_q [\log p(y_{ab} | z_{a-b}, z_{a-b}, \beta)] = E_q \left[ \frac{1}{c h(i)} \sum_{(a,b) \in h(i)} E_q [\log p(y_{ab} | z_{a-b}, z_{a-b}, \beta)] \right]. \] \[ \text{[S7]} \]

Under set-based sampling, the stochastic gradients of the ELBO are as follows:

\[ \frac{\partial g'_t}{\partial \beta_k} = \alpha_k + \frac{1}{c h(i)} \sum_{(a,b) \in h(i)} \phi_{a-b,k} - \gamma_{a,k}^{-1}, \] \[ \text{[S8]} \]

\[ \frac{\partial g'_t}{\partial \phi_{a-b,k}} = \eta_k + \frac{1}{c h(i)} \phi_{a-b,k} \phi_{a-b,k} y_{ab,i} - \lambda_{a,k}^{-1}. \] \[ \text{[S9]} \]

where \( y_{ab,0} = y_{ab} \), and \( y_{ab,1} = 1 - y_{ab} \). The global steps are the same as in Eq. S6.

**The Local Step.** The local step optimizes the interaction parameters \( \phi \) with respect to a subsample of the network. Recall that there is an interaction membership parameter for each node pair \( \phi_{a-b} \) and \( \phi_{a-b} \) —— representing the posterior approximation of which communities are active in determining whether there is a link. We optimize these parameters in parallel. (We will discuss an alternative local step optimization for the interaction parameters in Link Sampling.) The update for \( \phi_{a-b} \) given \( y_{ab} \) is as follows:

\[ \phi'_{a-b,k} y_{ab} = 0 \exp \left\{ E_q [\log \theta_{a,k}] + \phi_{a-b,k}^{-1} \right\} \] \[ + \left( 1 - \phi_{a-b,k}^{-1} \right) \log(1 - \epsilon) \] \[ \phi_{a-b,k} y_{ab} = 1 \exp \left\{ E_q [\log \theta_{a,k}] + \phi_{a-b,k}^{-1} \right\} \] \[ + \left( 1 - \phi_{a-b,k}^{-1} \right) \log(1 - \epsilon) \]. \[ \text{[S10]} \]

The updates for \( \phi_{a-b} \) are symmetric. Thus, we iteratively update \( \phi_{a-b,k} \) using \( \phi_{a-b,k}^{-1} \) and \( \phi_{a-b,k} \) using \( \phi_{a-b,k}^{-1} \) until convergence. In Eq. S10, \( i \) counts the iterations within the local step. This is natural gradient ascent with a step size of 1.

**Sampling Strategies.** Our algorithm is flexible about how the subset of node pairs \( S \) is sampled, as long as the expectation of the stochastic gradient is equal to the true gradient. We can choose the distribution over pairs to sample from independently or choose the distribution over sets. There are several options.

- **Random pair sampling.** The simplest method is to sample node pairs uniformly at random. This method is an instance of independent set sampling, with \( g(a, b) \) (used in Eq. S3) equal to \( N(N-1)/2 \).

- **Random node sampling.** This method focuses on local neighborhoods of the network. A set consists of all of the pairs that involve one of the \( N \) nodes. At each iteration, we sample a set uniformly at random from the \( N \) sets, so \( h(t) = 1/N \). Because each pair involves two nodes, each link or nonlink appears in two sets and so \( c = 2 \). Following Eq. S8 and Eq. S9, we compute the stochastic gradients from a sampled node \( a \) as follows:

\[ \frac{\partial g'_t}{\partial \beta_k} = \alpha_k + \frac{N}{2} \sum_{(a,b)} \phi_{a-b,k} - \gamma_{a,k}^{-1}, \] \[ \text{[S11]} \]

\[ \frac{\partial g'_t}{\partial \phi_{a-b,k}} = \eta_k + \frac{N}{2} \sum_{(a,b)} \phi_{a-b,k} \phi_{a-b,k} y_{ab,i} - \lambda_{a,k}^{-1}, \] \[ \text{[S12]} \]

where \( y_{ab,0} = y_{ab} \), and \( y_{ab,1} = 1 - y_{ab} \). In practice, we sample a “mini-batch” of nodes per update, to reduce noise.

**Informative set sampling.** The idea behind this method is to sample a set of pairs around each node with a bias toward pairs that help estimation. This is a type of set-based sampling.

For each node \( a \), we define an “informative set” consisting of all of its links and a small number of nonlinks. In our experiments, we chose nonlinks to nodes that are at most \( h \) hops from the node \( a \). (We set \( h = 2 \).) Such nodes may be more relevant to estimating the communities of node \( a \) (9). For each node, we also define \( m \) “noninformative sets” that partition its remaining nonlinks. Because the number of nonlinks associated with each node is large, dividing them into many sets allows the computation in each iteration to be fast. At each iteration, we select a node uniformly at random from the \( N \) nodes and choose the informative set with high probability by flipping a biased coin. Otherwise, with low probability, we select one of the \( m \) noninformative sets of the selected node. To compute Eq. S7, we note that \( c = 2 \) and the distribution over sets is...
Note that we set $\xi \ll 1$. We describe additional sampling methods in ref 3.

**Link Sampling.** The above subsampling methods include the network nonlinks. Many real networks are sparse and only a small fraction of their node pairs are links (Table S2). As the number of nodes increases, subsampling nonlinks becomes increasingly inefficient. Here, we consider “link-based variational inference” and “link sampling,” a subsampling approach that involves only the links in the network. We develop this algorithm by assuming that a node’s nonlinks are explained by the same communities that a node exhibits while generating links.

We specify the variational family in a particular way to focus on the links. It differs from the family in Eq. S1 in the variational interaction parameters for the links. The new family specifies an interaction parameter for the joint distribution over the pair of node community indicators of each link. The interaction parameters for the nonlinks remain the same as in Eq. S1. We then constrain the nonlink interaction parameters of each node to equal the mean of the link interaction parameters of that node. In the following discussion, $\text{links}(a)$ is the set of links of node $a$ in the training set, and $\text{nonlinks}$ are the set of all links in the training set. We define the sets for nonlinks similarly.

In particular, we use the following family in link-based variational inference,

$$q(\theta, z, \beta) = \prod_{n=1}^{N} q(\theta_n | \gamma_n) \prod_{(i,j) \in \text{links}} q(z_{i \rightarrow j}, z_{j \rightarrow i} | \phi_{ij}) \prod_{(i,j) \in \text{nonlinks}} q(z_{i \rightarrow j} | \phi_{ij})$$

We constrain the interaction parameters of each nonlink $(i, m)$ of a node $i$,

$$\phi_{i \rightarrow m, k} = \frac{\sum_{(j,l) \in \text{links}(i)} \phi_{ij}^{kl}}{d_{i}} + \frac{\sum_{(j,l) \in \text{nonlinks}(i)} \phi_{ij}^{kl}}{d_{i}} = \bar{\phi}_{i, k}.$$

where $d_{i}$ is the degree of node $i$ in the training set.

The simplification in Eq. S15 arises because $\sum_{k=1}^{N} \phi_{ij}^{kl} = 0$. When $k \neq l$, the community strength parameters are the nondiagonal entries of the block model, each set to 0 by our modeling assumption of assortativity, when $k \neq l$, $\phi_{ij}^{kl} \approx \exp(-\infty)$. Notice that because $\sum_{k=1}^{N} \bar{\phi}_{i, k} = 1$, $\bar{\phi}_{i, k}$ is normalized.

The ELBO in the link-based variational inference is a function of the variational parameters $(\bar{\phi}_{\text{links}}, \bar{\phi}, \gamma, \lambda)$. The $\bar{\phi}_{\text{links}}$ are the $M \times K$ matrix of interaction parameters defined over the links, where $M$ is the number of links in the training set. The $\bar{\phi}$ are the $N \times K$ matrix of the mean interaction parameters. We can compute the optimal $\phi_{ab}$, given a link $y_{ab} = 1$, while fixing the other parameters:

$$\phi_{ab}^{kl} \propto \exp\{E_q \log \theta_{ab} + E_q \log \theta_{ab} + E_q \log \beta_k\}.$$

The natural gradient of the ELBO with respect to the node’s community memberships is as follows:

$$\delta \gamma_{a,k} = \alpha_k + \sum_{(a,b) \in \text{links}(a)} \phi_{ab}^{kl} + \sum_{(a,b) \in \text{nonlinks}(a)} \phi_{a-b,k} - \gamma_{a,k}^{-1}$$

$$= \alpha_k + \sum_{(a,b) \in \text{links}(a)} \phi_{ab}^{kl} + c_a \gamma_{a,k} - \gamma_{a,k}^{-1},$$

where $c_a$ is the number of nonlinks of node $a$ in the training set. The natural gradient of the ELBO with respect to the community strengths is as follows:

$$\delta \gamma_{k,0} = \eta_0 + \sum_{(a,b) \in \text{links}} \phi_{ab}^{kl} - \gamma_{k,0}^{-1}$$

$$\delta \gamma_{k,1} = \eta_1 + \sum_{(a,b) \in \text{nonlinks}} \phi_{a-b,k} - \gamma_{k,1}^{-1}.$$

We can rewrite Eq. S18 as a function of only the link interaction parameters using the following:

$$\sum_{(a,b) \in \text{nonlinks}} \phi_{a-b,k} \bar{\phi}_{h,k} = \sum_{(a,b) \in \text{nonlinks}} \left( \frac{\sum_n \bar{\phi}_{n,k} \bar{\phi}_{n,k} - \sum_n \bar{\phi}_{n,k}^2}{2} \right).$$

We have expressed the natural gradients of the community memberships and community strengths as a function of $\phi_{\text{links}}$ and $\bar{\phi}$. We now describe a SVI algorithm that iterates only over the links.

Our link subsampling method extends random node sampling. The structure of the algorithm is similar to the general SVI algorithm, with a subsampling step, local steps, and global steps. At each iteration, we sample a node uniformly at random and observe all of its training links. In practice, we sample a minibatch of nodes. In the local step, we iterate over the links and compute the optimal link interaction parameters using Eq. S16. We then compute the mean interaction parameters of the sampled nodes using Eq. S15.

As with the previous sampling methods, we consider the stochastic optimization of the global community strengths $\lambda$ and the global community memberships $\gamma$. Previously, we obtained community membership gradients with respect to the entire vector $\gamma$ of dimension $N \times K$.

In link sampling, we optimize the community memberships of each node separately, using distinct learning rates. Furthermore, when we sample a node, we observe all links of a sampled node in the training set. We can therefore update the community memberships of the sampled node using the complete natural gradients in Eq. S17.

Because many networks are sparse, including the real datasets and the synthetic networks analyzed in the article, the link sampling algorithm scales to such networks even when the minibatch in each iteration is the set of all links in the training set. In this case, the natural gradients in Eq. S17 and Eq. S18 are used in the global step.

In our study on synthetic networks, we set the minibatch to the entire set of links and used a learning rate of 1. This leads to good convergence of the variational objective (Fig. S1). Furthermore, we rescaled $\gamma$ during an initial phase as follows:

$$\gamma_{a,k} = \gamma_{a,k} + \frac{\sum_{(i,j) \in \text{links}} \phi_{ij}^{kl}}{\sum_{(i,j) \in \text{links}} \phi_{ij}^{kl}}$$

The rescaling of $\gamma$ in Eq. S20 ensures that each community makes an equal contribution to the observations. This avoids small communities with high community strengths and unused communities during the early iterations. The initial phase is run until the expected log likelihood on a held-out set of node pairs no
longer improves. At this point, the inference continues without the scaling in Eq. S20 until the algorithm converges. (This can be interpreted as a form of annealing, a technique that is sometimes used in variational inference.)

As we demonstrate in the empirical study on synthetic networks, the SVI algorithm with link sampling recovers true communities with high accuracy, and scales to networks with millions of nodes.

Further subsampling can be applied to improve the efficiency of the SVI algorithm with link sampling. For instance, we can apply informative set sampling to the links. We maintain two dynamic sets of links: links whose corresponding interaction parameters have “converged” and links that have not converged. Each iteration, we sample links with a bias toward links that have not converged. (See Eq. S13.)

Setting the Number of Communities and Initializing Parameters. SVI requires initial settings of the global variational parameters. There are many ways to initialize these parameters. We set the community strength parameters $\gamma$ from “false observations” by dividing the links and nodes equally among the communities and adding a small random offset drawn from a Gamma distribution with mean 1. We initialize the community memberships $\gamma$ randomly in our empirical study on real and synthetic networks. Alternatively, we can initialize the $\gamma$ using an “initialization algorithm” that we describe below.

The initialization algorithm provides a decomposition of the network into overlapping communities that can be used to initialize the community memberships $\gamma$ and set the number of communities in the SVI algorithm. These communities are a good start, but it significantly improves as we run the SVI algorithm.

The pseudocode of our fast, scalable initialization algorithm for estimating the number of communities $K$ is enumerated below.

1. Initialize variational parameters of MMSB model $M$.
   - $M$ has $N$ nodes and $N$ communities.
   - Initialize $\gamma = \{\gamma_n\}^N_{n=1}$ randomly.
   - Assign each node $n$ to its own community $n$ by adding a small positive weight to $\gamma_{n,n}$.
   - Keep only the top $r$ communities of each node.

2. For each link $(a, b)$ in the training set,
   - Let $t_a$ and $t_b$ be the top communities of nodes $a$ and $b$.
   - Set $\gamma_{a,t_a} \leftarrow \gamma_{a,t_a} + 1$; $\gamma_{b,t_b} \leftarrow \gamma_{b,t_b} + 1$.

3. Recompute the top $r$ communities of each node.

4. Repeat steps 2, 3 for log $N$ iterations.

5. For each link $(a, b)$ in the training set,
   - Assign nodes $a$ and $b$ to community $k$ if the approximate posterior probability $p(z_{a=b} = z_{a=b} = k | y, M) > 0.5$.

6. Return the overlapping communities and their cardinality.

The initialization algorithm approximates a batch variational inference algorithm for a subclass of the MMSB where the community strengths $\beta$ are set to 1. This algorithm is fast: It completes in minutes on networks with millions of nodes and thousands of communities. In simulations, the algorithm frequently finds the number of communities reasonably close to the ground truth number. (See the empirical study on synthetic networks.)

The algorithm begins by assigning each node to its own community. It then computes the community memberships $\gamma$ for all nodes while maintaining only the top $r$ communities with each node. (We set $r = 5$ in all our experiments.)

Under the restricted model, with community strengths set to 1 and nodes initialized to their own community, the local step for a link in Eq. S10 dictates that the optimal community indicator for node $a$ is the dominant community of node $b$, and vice versa. This amounts to an exchange of the dominant community memberships of the nodes and is computed in $O(1)$ time by maintaining the peak communities of nodes. Such exchanges bear similarities to the label propagation steps in ref 10.

The algorithm terminates after exactly log $N$ iterations, where $N$ is the number of nodes. This stopping criteria is reasonable under the assumption of “small-world” behavior, where the average path length in the network grows proportional to log $N$ (11).

When the initialization algorithm terminates, it writes a list of communities. Each community consists of a list of nodes, and nodes can appear in multiple communities. A node is added to community $k$ if it is adjacent to at least one link whose approximate posterior probability of belonging to community $k$ is greater than a high threshold. We can use the list of communities to initialize the $\gamma$ for the SVI algorithm. For example, we can initialize the memberships of a node randomly but with a greater weight on the community assignments from the initialization algorithm. The number of communities, i.e., the number of ways in which links are colored, gives us the input $K$ for the SVI algorithm.

We note again that the initialization algorithm provides us with an optional starting point for the SVI algorithm and an estimate of the number of communities in the observed data.

Computational Complexity. The local step of the SVI algorithm can be computed in $O(SK)$ operations, where $S$ is the number of node pairs sampled in each iteration and $K$ is the number of communities. Due to the assortativity assumptions in our model, the local step is not quadratic in $K$ as is typical for the MMSB (2, 3). The time for the global step of the SVI algorithm is $O(NK)$ operations per iteration, where $N$ is the number of nodes. To avoid updating all nodes in the network, we can maintain a distinct learning rate for each node. In a given iteration, we skip updating the community membership parameters and learning rates of nodes not in the minibatch. The sequence of positive step sizes used in updating a node’s membership parameter continue to satisfy the Robbins–Monro conditions (7). This improves the time for the global step to $O(nK)$ operations per iteration, where $n$ is the number of nodes in the minibatch.

In the SVI algorithm with link sampling, with the minibatch set to all links, the computational complexity is $O(MK + NK)$ operations per iteration, where $M$ is the number of links. The SVI algorithm with link sampling does not require subsampling nonlinks and converges much faster than other subsampling methods.

Open-Source Software. We implemented the SVI algorithm and the various subsampling variants in C++. (Our software is available at https://github.com/premgopalan/svini.) The software takes as input a text file of undirected links, the number of nodes, the type of subsampling method, and optionally, the hyperparameter values and the number of communities. The software generates as output the list of discovered overlapping communities, the fitted model, the computed log likelihood on various held-out sets, and Graph Modeling Language (GML) format files for visualizing the communities.

Empirical Study on Real-World Networks

In this section, we describe the details of the empirical study on real-world networks. We ran the SVI algorithm with informative set sampling on the real networks in Table S2. The input to the SVI algorithm is a list of links and the number of communities. We preprocessed the networks to associate each node with “informative” and “noninformative” sets of node pairs.

Assessing Convergence on the Training Set. We measure convergence of the SVI algorithm by computing the link prediction accuracy on a held-out set of node pairs. In our experiments on real networks, we set aside two validation sets and a test set, each
having \( h \% \) of the network links and an equal number of non-links. In the experiments on real networks, we set \( h = 5\% \). The links and nonlinks are chosen from the network uniformly at random. We use the validation sets to assess convergence, choose learning parameters, and study the sensitivity to the number of communities.

A “50%–links” validation set poorly represents the severe class imbalance between links and nonlinks in real-world networks. For example, links form only 0.0039% of the node pairs in the arXiv network (12) listed in Table S2. On the other hand, a validation set matching the network sparsity would have too few links. We address the class imbalance by computing the “validation log likelihood at network sparsity.” This quantity is computed by reweighting the average link and nonlink log likelihood (estimated from the 50% links validation set) by their respective proportions in the network.

We stop training when the average change in expected validation log likelihood at network sparsity is less than 0.001% or if the expected validation log likelihood at network sparsity no longer increases.

Under the MMSB, we approximate the predictive distribution using point estimates of the posterior community memberships of nodes \( \theta \) and the posterior community strengths \( \beta \); these point estimates are computed as the mean of the variational posterior parameters \( \gamma \) and \( \lambda \), respectively. The estimated predictive distribution of a held-out node pair \( y_{ab} \), is as follows:

\[
 p(y_{ab} | y_{\text{observed}}) \approx \sum_{z_{a-b}} \sum_{z_{a-b}} p(y_{ab} | z_{a-b}, z_{a-b}) p(z_{a-b} | \gamma_{a-b}) p(z_{a-b} | \lambda_{a-b}).
\]  

[S21]

It is straightforward to show that Eq. S21 is a valid approximation. We then evaluate the log probability of the node pairs in the held-out set under this distribution. Fig. S2 shows the convergence of the “perplexity” at network sparsity on a validation set. Results are shown for the arXiv network (12) and the Google network (13). Perplexity is defined using the average predictive log likelihood of a held-out set of node pairs \( H \),

\[
 \text{perplexity}(H) = \exp \left\{ \frac{\sum_{a,b \neq H} \log p(y_{ab} | y_{\text{observed}})}{|H|} \right\}.
\]  

[S22]

Perplexity is a measure of model fitness (lower numbers are better). Fig. S2 shows that the SVI algorithm with informative set sampling can approximate the posterior distribution on large networks in several hours. Notice that the perplexity values are small in magnitude. This is because we compute the validation log likelihood at network sparsity. The model can predict a large fraction of the nonlinks with high accuracy, and is not “surprised” by them.

**Model Selection.** As with many probabilistic models of community detection, the MMSB requires setting the number of communities. In our empirical study, we addressed this model selection problem in two ways. One was by evaluating the predictive performance of the model for varying numbers of communities. We held out a set of node pairs and computed the average predictive log likelihood, as described above. A better model will assign a higher probability to the held-out set. This reflects a predictive approach to model selection, and has good statistical properties (14). (We note that nonprobabilistic methods for detecting overlapping communities usually cannot provide a mechan-ism for predicting unseen pieces of the network.) Fig. S3 shows the sensitivity of the MMSB to the number of communities on the arXiv network (12) and the Google network (13). A second way was to set the number of communities to the estimate from

Having our initialization algorithm. We used this second approach in our empirical study on synthetic networks.

**Comparison with the Stochastic Blockmodel.** We compared the model fitness of the MMSB to the stochastic blockmodel (15) on real-world networks. The stochastic block model attaches a single community to each node. We consider the following constrained stochastic blockmodel, with \( K \) communities, in a full Bayesian setting (16, 17). The modeling assumptions are captured in the following generative process:

1. For each community \( k \),
   (a) Draw intracommunity strengths \( \beta_k \sim \text{Beta}(\eta) \).
2. Draw the intercommunity strength \( \beta' \sim \text{Beta}(\eta') \).
3. Draw the node memberships \( \theta \sim \text{Dirichlet}(\alpha) \).
4. For each node \( i \):
   (a) Draw a community indicator \( z_i \sim \theta \).
5. For each pair of nodes \( i \neq j \), where \( i < j \):
   (a) Draw the connection between them from
   \[
   p(y_{ij} = 1 | z_i, z_j, \beta) = \begin{cases} 
\beta & \text{if } z_i = z_j \\
\beta' & \text{if } z_i \neq z_j.
\end{cases}
\]  

Unlike the MMSB of ref. 3, the single-membership model of Eq. S23 cannot explain all links as arising from shared memberships; hence, it must learn the intercommunity strength \( \beta' \) from the data. Our model is a generalization of ref 16.

We derived a scalable SVI algorithm for the model in Eq. S23 by treating all hidden variables, including the community indicators, as global. This is necessary because the community indicators associated with each node are not local to an observation. Therefore, the variational parameters, including those for the community indicators, were updated using noisy natural gradients in the global step. We note that the subsampling methods discussed earlier, with the exception of link sampling, apply to the single-membership model.

In Fig. S3, we compared the predictive performance of the MMSB to the stochastic blockmodel on the arXiv network and the Google network. We fit both models using the SVI algorithm with informative set sampling. Fig. S3 shows that the mixed-membership model demonstrates better predictive performance than the analogous single-membership model of Eq. S23 over a range of the number of communities.

**Hyperparameters and Learning Parameters.** SVI requires setting hyperparameters of the model and learning rates of the algorithm. We set the node membership forgetting rate (\( \kappa \)) to 0.5 and the community strength forgetting rate to 0.9. We set the delay \( \tau_0 = 1024 \). We set Dirichlet hyperparameters \( \alpha = \frac{1}{K} \), where \( K \) is the number of communities. On real networks, the prior on the community strengths was set using a uniform assignment of links and nodes to communities. We set the probability of a link when nodes assume different communities, \( \varepsilon \), to a low value of \( 10^{-30} \).

This reflects our assortativity assumption that links arise from similarity in communities between a pair of nodes.

**Empirical Study on Synthetic Networks**

The goal of the study on synthetic networks is to assess the accuracy of the SVI algorithm and compare with other scalable methods in the research literature. We want the synthetic networks to match the properties of real networks—skewed community and node degree distributions (18), significant community overlap (19, 20), and a large fraction of nodes with multiple memberships (20). We ran experiments to evaluate the performance of the algorithms on benchmark networks with and without “noisy” links. Notice that our significant community overlap requirement naturally avoids well-separated clusters. The inclusion of noisy links tests the algorithm’s ability to identify overlapping com-
munities even when a significant fraction of a node’s links are to nodes sharing no communities.

For the experiments on networks without noise, we generated 20 Lancichinetti–Fortunato–Radicchi (LFR) benchmark networks (21) varying in size from \( N = 1000 \) to \( N = 1,000,000 \) nodes. One-half of the nodes in each network have memberships in \( m = 4 \) communities. We set the average degree of nodes as \( 15 \times m \), similar to the experiments in ref. 22. The LFR benchmarks give the node degrees and community sizes power laws; the degree distribution and community size distribution exponents were set to the default values of 2.0 and 1.0, respectively. Research on scale-free networks (23) have assumed the maximum degree to vary as \( k_{\text{max}} \sim N^\alpha \), where \( \alpha \) is the power law exponent for node degrees. We set \( k_{\text{max}} = \sqrt{N} \). We varied the minimum and maximum community sizes as \( \left( 20 \frac{N}{1000}, 50 \frac{N}{1000} \right) \). However, because community sizes are typically small (13), we set the minimum and maximum community sizes when \( N = 1,000,000 \) nodes to (2000, 5000). These settings result in about \( \sim 750 \) ground truth communities when \( N = 1,000,000 \) and \( \sim 30 \) communities when \( N = 1,000 \). Finally, we set the “mixing parameter” \( \mu \) (21) to 0 in our experiments on networks without noise. The mixing parameter is the fraction of a node’s links that connect to nodes sharing no communities.

On each network, we ran the following algorithms:

1. The COPRA label propagation algorithm (10).
2. The INFOMAP algorithm based on flow compression (24).
3. The MOSES seed expansion algorithm (22).
4. The Poisson expectation-maximization (EM) algorithm (the Poisson community model, fit with EM) (25).
5. The OSLOM algorithm for finding statistically significant communities (26).
7. The Link clustering algorithm (20).

For the experiments on networks with noisy links, we varied \( \mu \) in steps of 0.2 from 0 to 0.8. We fixed the number of nodes at 10,000, and kept the other settings the same as the preceding experiment. We generated 25 LFR benchmark networks and included only the candidate algorithms that successfully scaled to 1,000,000 nodes in the preceding experiment.

We used the author’s source code for all algorithms. We ran the SVI algorithm with the link sampling method. For each run, we measured the normalized mutual information (NMI) (21) between the inferred community structure and the true community structure. For the algorithms that find communities at various resolutions—Clique percolation, Link clustering, and COPRA—we varied the parameters as described below, and kept the best NMI score.

For the SVI and the Poisson EM algorithm, we ran the algorithms until convergence on networks with up to 100,000 nodes. We measured convergence of the SVI algorithm with link sampling using the average change in average validation log likelihood at network sparsity, as we did with the experiments on real networks. However, since our goal is to assess the accuracy in recovering ground truth communities, we set aside only a single validation set of node pairs, having \( 1\% \) of network links and an equal number of nonlinks. We gave all algorithms, except the SVI algorithm, the complete synthetic network as input.

On the million node networks, the SVI and the Poisson EM algorithm can take a long time for convergence in likelihood, whereas their NMI scores have typically “converged” quickly. One reason for this is the large number of links (~54 million links) in these synthetic networks. We instrumented the author’s source code for the Poisson EM algorithm and the SVI algorithm to periodically report the accuracy scores when provided with ground truth communities. We gave both algorithms a computational budget of 24 h and recorded the NMI scores attained by them. The Poisson EM algorithm’s NMI score had typically “converged” at this point, even if the likelihood did not. (We note that in other applications of EM, such as probabilistic latent semantic indexing, “early stopping” is an effective form of regularization.)

Table S3 shows the NMI results on the networks without noise. Some algorithms could not scale to one million node networks. The four that did were the Poisson EM, the SVI, the COPRA, and the INFOMAP algorithms. The SVI algorithm performs better than the COPRA and the INFOMAP algorithms and is as accurate as the Poisson EM algorithm on the one million node network. On smaller networks, the SVI algorithm performs as well as the Poisson EM algorithm; it performs second to Clique percolation on the one thousand node networks. However, the Clique percolation algorithm does not scale beyond the 10,000 node networks.

Fig. S4 shows the NMI results on the networks with noisy links. We find that the SVI algorithm performs better than two of the three other scalable alternatives—the COPRA and the INFOMAP algorithms—and is as accurate as the Poisson EM algorithm.

Hyperparameters and Learning Parameters. We set the number of communities \( K \) of the SVI algorithm with link sampling to the value chosen by the initialization algorithm. We provided the same \( K \) to the Poisson EM algorithm (25). We set the minibatch for the link sampling method to set of all training links, and set the learning rate to 1. Other hyperparameters of the SVI algorithm were set to the same values as our experiments on real networks.

Algorithm Settings for the LFR Experiments. The Clique percolation algorithm identifies communities from a series of adjacent k-cliques (19). In our experiments, we varied \( k \) from 4 to 7, a typical range for LFR experiments (10, 22). The Link clustering algorithm defines a similarity function over nodes sharing a link, and uses hierarchical clustering to find hierarchical community structures (20). Because the dendrogram can be partitioned in multiple ways, the algorithm uses a measure of the quality of a link partition, called the partition density \( D \). We varied \( D \) from 0.1 to 0.4—the range we found to be best—in steps of 0.1. The COPRA algorithm is a fast heuristic based on label propagation and includes a overlap parameter that we varied from 2 to 10, in steps of 2. This is a typical range (10). The OSLOM (26), MOSES (22), and the INFOMAP (24) algorithms were run with parameters set to default values.

The author’s software for most of the algorithms compare with generate “community assignments,” the discovered mapping between nodes and communities. The mapping is used to compute the accuracy when given the ground truth communities. We extended both the SVI and the Poisson EM algorithm to generate the community assignment files. In both algorithms, we assigned a link to a community if the approximate posterior probability of link assignment to a community exceeded a threshold \( t \). We took the best NMI values obtained from thresholds \( r = 0.5 \) and \( t = 0.9 \). For the experiments on networks without noise, we assigned each node associated with a link to the same community as the link. For the experiments with noisy links, we required at least three links of a node to be assigned to a community before assigning the node to that community. We added this setting to both algorithms to control sensitivity to noise. [Notice in Fig. S4 that both algorithms continue to show a high accuracy on networks without noise (\( \mu = 0 \)) with the threshold set to three links.]
The convergence of held-out perplexity on an LFR synthetic network (1) with 1,000,000 nodes and ~54,000,000 links. The SVI algorithm was run with link sampling. The network has 750 ground truth overlapping communities, and 50% of the nodes have memberships in four communities. The degree distribution and the community size distribution follow power laws.


Fig. S1. The convergence of held-out perplexity on an LFR synthetic network (1) with 1,000,000 nodes and ~54,000,000 links. The SVI algorithm was run with link sampling. The network has 750 ground truth overlapping communities, and 50% of the nodes have memberships in four communities. The degree distribution and the community size distribution follow power laws.

Fig. S2. The convergence of the held-out perplexity at network sparsity on the arXiv network (1) (Left) and the Google network (2) (Right). The SVI algorithm was run with informative set sampling. The number of communities $K$ was set to 1,000.


Fig. S3. Perplexity results on the arXiv network (1) (Left) and the Google network (2) (Right). The mixed-membership model outperforms the single-membership model. We fit both models using the SVI algorithm with informative set sampling.

Fig. S4. The stochastic inference algorithm (SVI) with link sampling outperforms COPRA (COP) (1) and INFOMAP (INF) (2) and is as accurate as the Poisson EM algorithm (POI) (3) in discovering ground truth communities in 25 LFR benchmark networks with noisy links. Each panel shows the performance of the algorithms on five replications of the random network generated with 10,000 nodes and a fixed mixing parameter (4). The mixing parameter is the fraction of a node’s links that connect to nodes sharing no communities. From Left to Right, the panels correspond to increasing noise.

Table S1. Top 10 articles in the arXiv network (1) by estimated bridgeness (2)

<table>
<thead>
<tr>
<th>Title</th>
<th>No. citations</th>
<th>Estimated bridgeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maps of dust infrared emission for use in estimation of reddening and cosmic microwave background radiation foregrounds (3)</td>
<td>5,946</td>
<td>2,893.7</td>
</tr>
<tr>
<td>First-year Wilkinson microwave anisotropy probe (WMAP) observations: Determination of cosmological parameters (4)</td>
<td>5,707</td>
<td>2,270.7</td>
</tr>
<tr>
<td>Three-year Wilkinson microwave anisotropy probe (WMAP) observations: Implications for cosmology (5)</td>
<td>4,488</td>
<td>1,907.1</td>
</tr>
<tr>
<td>The cosmological parameters 2006 (7)</td>
<td>2,896</td>
<td>1,882.9</td>
</tr>
<tr>
<td>Five-year Wilkinson microwave anisotropy probe (WMAP) observations: Cosmological interpretation (8)</td>
<td>2,804</td>
<td>1,485.9</td>
</tr>
<tr>
<td>A large mass hierarchy from a small extra dimension (9)</td>
<td>3,644</td>
<td>1,426.7</td>
</tr>
<tr>
<td>The large N limit of superconformal field theories and supergravity (10)</td>
<td>3,914</td>
<td>1,378.4</td>
</tr>
<tr>
<td>An alternative to compactification (11)</td>
<td>2,803</td>
<td>1,275.8</td>
</tr>
</tbody>
</table>

Notice that some articles have higher bridgeness but a smaller citation count than others.
Table S2. Real-world networks analyzed in the article and SI Text

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No. of nodes</th>
<th>No. of links</th>
<th>% links</th>
<th>Type</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>arXiv</td>
<td>576,000</td>
<td>6,640,000</td>
<td>0.0039%</td>
<td>Citation</td>
<td>Ref. 1</td>
</tr>
<tr>
<td>Google</td>
<td>875,000</td>
<td>4,320,000</td>
<td>0.0011%</td>
<td>Hyperlink</td>
<td>Ref. 2</td>
</tr>
<tr>
<td>US patents</td>
<td>3,700,000</td>
<td>16,500,000</td>
<td>0.00023%</td>
<td>Citation</td>
<td>Ref. 3</td>
</tr>
</tbody>
</table>


Table S3. Accuracy results on 20 LFR benchmark networks (1) measured using normalized mutual information (1)

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Replication</th>
<th>SVI</th>
<th>COPRA (2)</th>
<th>INFOMAP (3)</th>
<th>MOSES (4)</th>
<th>POISSON (5)</th>
<th>OSLOM (6)</th>
<th>CLIQUE (7)</th>
<th>LC (8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000</td>
<td>1</td>
<td>0.58</td>
<td>0.55</td>
<td>0.38</td>
<td>0.47</td>
<td>0.62</td>
<td>0.44</td>
<td>0.93</td>
<td>0.15</td>
</tr>
<tr>
<td>1,000</td>
<td>2</td>
<td>0.77</td>
<td>0.45</td>
<td>0.36</td>
<td>0.49</td>
<td>0.77</td>
<td>0.41</td>
<td>0.85</td>
<td>0.16</td>
</tr>
<tr>
<td>1,000</td>
<td>3</td>
<td>0.66</td>
<td>0.46</td>
<td>0.36</td>
<td>0.53</td>
<td>0.66</td>
<td>0.49</td>
<td>0.96</td>
<td>0.17</td>
</tr>
<tr>
<td>1,000</td>
<td>4</td>
<td>0.63</td>
<td>0.17</td>
<td>0.38</td>
<td>0.52</td>
<td>0.62</td>
<td>0.46</td>
<td>0.78</td>
<td>0.15</td>
</tr>
<tr>
<td>1,000</td>
<td>5</td>
<td>0.76</td>
<td>0.39</td>
<td>0.35</td>
<td>0.55</td>
<td>0.75</td>
<td>0.48</td>
<td>0.85</td>
<td>0.20</td>
</tr>
<tr>
<td>10,000</td>
<td>1</td>
<td>0.90</td>
<td>0.28</td>
<td>0.35</td>
<td>0.56</td>
<td>0.85</td>
<td>0.18</td>
<td>0.22</td>
<td>0.01</td>
</tr>
<tr>
<td>10,000</td>
<td>2</td>
<td>0.90</td>
<td>0.28</td>
<td>0.32</td>
<td>0.55</td>
<td>0.88</td>
<td>0.16</td>
<td>0.13</td>
<td>0.01</td>
</tr>
<tr>
<td>10,000</td>
<td>3</td>
<td>0.82</td>
<td>0.07</td>
<td>0.36</td>
<td>0.54</td>
<td>0.78</td>
<td>0.19</td>
<td>0.23</td>
<td>0.01</td>
</tr>
<tr>
<td>10,000</td>
<td>4</td>
<td>0.86</td>
<td>0.61</td>
<td>0.44</td>
<td>0.54</td>
<td>0.82</td>
<td>0.17</td>
<td>—</td>
<td>0.02</td>
</tr>
<tr>
<td>10,000</td>
<td>5</td>
<td>0.89</td>
<td>0.62</td>
<td>0.40</td>
<td>0.56</td>
<td>0.86</td>
<td>0.17</td>
<td>—</td>
<td>0.00</td>
</tr>
<tr>
<td>100,000</td>
<td>1</td>
<td>0.82</td>
<td>0.57</td>
<td>0.34</td>
<td>0.35</td>
<td>0.85</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>100,000</td>
<td>2</td>
<td>0.83</td>
<td>0.44</td>
<td>0.33</td>
<td>0.33</td>
<td>0.81</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>100,000</td>
<td>3</td>
<td>0.81</td>
<td>0.50</td>
<td>0.35</td>
<td>0.34</td>
<td>0.81</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>100,000</td>
<td>4</td>
<td>0.82</td>
<td>0.43</td>
<td>0.33</td>
<td>0.35</td>
<td>0.84</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>100,000</td>
<td>5</td>
<td>0.83</td>
<td>0.58</td>
<td>0.33</td>
<td>0.35</td>
<td>0.84</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1,000,000</td>
<td>1</td>
<td>0.76</td>
<td>0.52</td>
<td>0.22</td>
<td>—</td>
<td>0.76</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1,000,000</td>
<td>2</td>
<td>0.77</td>
<td>0.50</td>
<td>0.16</td>
<td>—</td>
<td>0.76</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1,000,000</td>
<td>3</td>
<td>0.78</td>
<td>0.53</td>
<td>0.17</td>
<td>—</td>
<td>0.77</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1,000,000</td>
<td>4</td>
<td>0.76</td>
<td>0.49</td>
<td>0.23</td>
<td>—</td>
<td>0.79</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1,000,000</td>
<td>5</td>
<td>0.77</td>
<td>0.51</td>
<td>0.14</td>
<td>—</td>
<td>0.77</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

The networks were generated with mixing parameter set to 0. The four algorithms that scale to a million nodes are the SVI algorithm, the Poisson EM algorithm (5), INFOMAP (3), and COPRA (2). The SVI algorithm performs better than INFOMAP and COPRA and is as accurate as the Poisson EM algorithm.