Journal of Machine Learning Research 17 (2016) 1-41 Submitted 4/15; Published 4/16

Latent Space Inference of Internet-Scale Networks

Qirong Ho* HOQIRONG@GMAIL.COM
Institute for Infocomm Research
*A*STAR*
Singapore 138632

Junming Yin* JUNMINGY@EMAIL.ARIZONA.EDU
Department of Management Information Systems
Eller College of Management
University of Arizona
Tucson, AZ 85721

Eric P. Xing EPXING@CS.CMU.EDU
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Editor: Jure Leskovec

Abstract

The rise of Internet-scale networks, such as web graphs and social media with hundreds of millions to billions of nodes, presents new scientific opportunities, such as overlapping community detection to discover the structure of the Internet, or to analyze trends in online social behavior. However, many existing probabilistic network models are difficult or impossible to deploy at these massive scales. We propose a scalable approach for modeling and inferring latent spaces in Internet-scale networks, with an eye towards overlapping community detection as a key application. By applying a succinct representation of networks as a bag of triangular motifs, developing a parsimonious statistical model, deriving an efficient stochastic variational inference algorithm, and implementing it as a distributed cluster program via the Petuum parameter server system, we demonstrate overlapping community detection on real networks with up to 100 million nodes and 1000 communities on 5 machines in under 40 hours. Compared to other state-of-the-art probabilistic network approaches, our method is several orders of magnitude faster, with competitive or improved accuracy at overlapping community detection.

Keywords: probabilistic network models, triangular modeling, stochastic variational inference, distributed computation, big data

1. Introduction

The rapid growth of the Internet, particularly the explosion of social media, has led to unprecedented increases in the volume of network data worldwide. Already, the Yahoo web graph collected in 2002 contains in excess of one billion URLs (Yahoo, 2013), the Facebook social network recently exceeded one billion users (Facebook, 2013), and numerous other social networks or online communities easily claim memberships in the millions of

*. Qirong Ho and Junming Yin contributed equally.

users (Leskovec and Krevl, 2014). To understand the structural and functional properties of massive networks, one important task in network analysis is unsupervised detection of community structure—a task that frequently occurs in social media and Internet studies (Palla et al., 2005; Newman, 2006; Prakash et al., 2010; Psorakis et al., 2011; Xie and Szymanski, 2012; Gopalan and Blei, 2013). Although there is no clear consensus in the literature on the definition of a network community, it is generally accepted that actors or nodes from the same community tend to interact more frequently with each other and share more common characteristics. In a social network, relevant characteristics might be "(people) going to the same school, social club, or workplace", while on the Internet, a characteristic might be "(websites) about a specific topic, such as politics, sports, or technology". It is well-understood that actors in a network can have multiple characteristics, resulting in multiple community memberships that usually overlap (Palla et al., 2005; Yang and Leskovec, 2012b); hence, it is widely accepted that overlapping communities provide a more accurate picture of network structure than disjoint communities (Xie et al., 2013).

To date, however, only a few overlapping community detection algorithms have been successfully applied to large graphs in excess of hundreds of millions of nodes (Prakash et al., 2010; Kang et al., 2011), and, to the best of our knowledge, none of them are based on a probabilistic formulation. Probabilistic methods are often more flexible in that they can serve as "building blocks" to more sophisticated models over additional non-network data sources—particularly text and feature data (Rosen-Zvi et al., 2004; Dietz et al., 2007; Nallapati et al., 2008; Chang and Blei, 2009; Balasubramanyan and Cohen, 2011; Ho et al., 2012a). They may also be extended to settings with multiple networks, e.g., the time-varying setting in which we are given snapshots of the same network at multiple time points (Fu et al., 2009; Ho et al., 2011), or to the nonparametric setting, which provides a principled way to automatically select the number of communities (Kemp et al., 2006; Ho et al., 2012a,b).

Despite some recent work showing that probabilistic methods can already be scaled up to networks with several million nodes (Gopalan and Blei, 2013), there are currently no probabilistic network models that have been reported to scale to hundreds of millions of nodes or more. In this paper, we develop a new probabilistic network model and accompanying inference algorithm for these massive scales. We take a latent space approach, in which each node is associated with an unobserved mixed-membership vector in the latent space that represents a mixture distribution over the multiple possible characteristics (roles or properties) it can have. The mixed-membership assumption can naturally capture the multi-faceted and heterogeneous nature of nodes in real networks—for example, actors in social networks tend to belong to multiple distinct social groups, depending on whom they are interacting with. This is in contrast to the single-membership assumption, where each node can only play a single latent role in its relationship with other nodes. We focus on probabilistic inference of mixed-membership vectors of individual nodes, taking as input the observed structural relation of Internet-scale networks with millions to hundreds of millions of nodes. The inferred mixed-membership position vectors can then be used for overlapping community detection. We demonstrate that our method can infer 1000 communities from a 101-million-node web graph in less than 40 hours using a small cluster of 5 machines, and that, on real-world networks with ground truth, our community recovery accuracy is competitive with or outperforms other scalable probabilistic models.

1.1 Challenges in Probabilistic Modeling of Large Networks

For a network with N nodes, its dense adjacency matrix representation contains $\Theta\left(N^2\right)$ elements. Although the network data itself can be stored in a sparse format (e.g., edge list or adjacency list), several popular probabilistic network models—such as mixed-membership stochastic blockmodels (Airoldi et al., 2008) and latent feature models (Miller et al., 2009)—associate latent variables to each of the N^2 elements in the adjacency matrix, essentially treating the network as non-sparse. Consequently, the inference algorithm naively requires $\Omega(N^2)$ computational complexity. This may still be acceptable for small networks, but is untenable for networks with hundreds of millions of nodes, as considered in this work.

A second difficulty is that larger networks have been observed to have more communities (Leskovec et al., 2005), so the number of hidden variables or parameters in the model often has to grow super-linearly in the number of communities K (in addition to N). Having super-linear growth in the state space can quickly render the inference algorithm computationally intractable, even at modest network scales. Such challenges have already been observed in the mixed-membership stochastic blockmodel (MMSB), which is infeasible for networks with $N \geq 100$ million nodes or $K \geq 100$ communities because of its $O(N^2K^2)$ per-iteration runtime. Recent work by Gopalan and Blei (2013) has brought the per-iteration inference complexity of MMSB down to $O(MK)$, where M is the number of edges. The resulting algorithm allows scalability of up to $N = 4$ million nodes and $K = 1000$ communities in about 1 week on a single machine. However, even that result remains about 1.4 orders of magnitude below our desired target of $N \geq 100$ million nodes.

1.2 Proposed Approach

From a scalability perspective, an ideal large-scale network inference algorithm would require only linear-time work in the number of nodes and communities. Towards this end, we take a different approach to data representation, model construction, and algorithm design that avoids the common edge-based representation of a network (e.g., adjacency matrices or adjacency lists), in favor of viewing the network in terms of *triangular motifs* (also known as triads, or subgraphs of size 3).

Our triangular motif representation captures certain edge patterns observed in node triples (i, j, k), such as the 2-edge triangle $i - j - k$ and the 3-edge triangle $i - j - k - i$. This is a hypergraph representation of the network, where every hyperedge is labeled with the edge pattern over (i, j, k)[1]. Hypergraph representations of networks have been studied in both the social networks (Ghoshal et al., 2009) and statistics (Stasi et al., 2014) literature, and there is an empirical evidence showing that triangles and higher-order subgraphs can substantially reveal the structure and communities within a network (Milo et al., 2002; Yang and Leskovec, 2012b). Moreover, this triangular representation has desirable properties for large-scale network analysis; in particular, it is possible to subsample triangular motifs in a manner that approximately preserves important network attributes such as the *clustering coefficient*, a popular measure of cluster strength.

In the following sections, we shall discuss how to build a scalable, linear-time network inference algorithm based on triangular motifs. We begin with the triangular data representation (Section 2), followed by statistical model design (Section 3), and then parallelizable

1. This is technically a 3-uniform hypergraph, where all hyperedges are associated with 3 nodes.

inference algorithm construction (Section 4). Finally, we implement the algorithm in the distributed, multi-machine cluster setting (Section 5), using a large-scale machine learning platform called Petuum (`www.petuum.org`). The resulting latent space inference algorithm can be used for overlapping community detection on networks with more than 100 million nodes and thousands of communities.

1.3 Related Work

Understanding network structure is highly important to a number of scientific domains, and each domain has developed its own unique tools to analyze the global and local properties of networks. Here, we provide a general overview of network analysis methods from the computer science and statistics literature, with a focus on how well they scale to very large networks with hundreds of millions of nodes or more.

In the general computer science literature, there has been an emphasis on achieving fast, linear-time overlapping community detection on medium-sized networks (usually millions of nodes). Examples include the GCE (Lee et al., 2010), Link (Ahn et al., 2010) and SLPA (Xie and Szymanski, 2012) algorithms. Although these algorithms are fast (less than one hour running time on 1 million nodes), their clustering strategies are not always based on a clearly defined optimization function or statistical model, and might be considered ad-hoc in a statistical or machine learning setting. For example, SLPA propagates community labels through the graph, and does not have an underlying statistical model or objective function. There is neither consensus on which approach works best nor on generalization guarantees to networks with different properties. Moreover, compared to a model-based approach, it is unclear how to incorporate additional network attributes in a principled manner, such as text or other meta-data associated with each node.

In statistics and machine learning, network algorithms are typically created by defining a formal model of the network, and deriving an inference or optimization algorithm to learn the model's parameters. Predominant statistical network models include exponential random graph models (ERGMs) (Morris et al., 2008; Hunter et al., 2008; Guo et al., 2007), stochastic blockmodels (Bickel et al., 2013; Anandkumar et al., 2014; Airoldi et al., 2008; Ho et al., 2011; Gopalan et al., 2012) and latent factor models (Miller et al., 2009; Handcock et al., 2007; Hoff et al., 2002). All of these models are popular for their ease of interpretability, but their inference algorithms generally require $\mathrm{O}(N^2)$ time due to how they model the adjacency matrix, making them patently unscalable to massive networks with $N \geq 100$ million. Exceptions to this trend include assortative MMSB (Gopalan et al., 2012; Gopalan and Blei, 2013), which applied stochastic gradient techniques to achieve linear runtime on the MMSB (Airoldi et al., 2008), and sparse block model (Parkkinen et al., 2009) with $\mathrm{O}(M)$ latent variables (M is the number of edges). Although these methods are scalable in their own right, they still treat *network edges* as the basis for the task of community detection. In this paper, we will argue that *triangular motifs* are a good, if not better, network representation for this task (Section 3), and design a network model based on these triangular motifs.

The natural language processing and information retrieval communities have extended the LDA topic model (Blei et al., 2003) to a variety of text-network models by exploring the links between documents. Examples include the relational topic model (Chang and

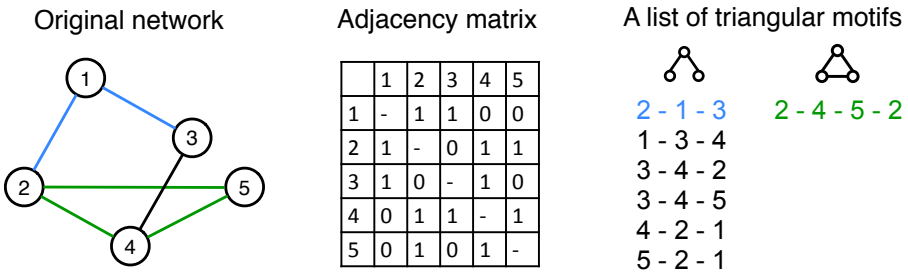
Blei, 2009), Link-PLSA-LDA (Nallapati et al., 2008), Block-LDA (Balasubramanyan and Cohen, 2011), the author-topic model (Rosen-Zvi et al., 2004), and the citation influence model (Dietz et al., 2007). While these models are scalable in that they require $O(M)$ work on the network data, most of them (except for the Block-LDA model) cannot perform network inference in the absence of text data because they do not use a stand-alone network model.

In the data mining and machine learning literature, matrix factorization methods provide a principled framework for decomposing relational data (of which networks are one type) into simpler "basis" components. Although the basis components are sometimes less interpretable than the probability distributions that arise from statistical models (e.g., the basis components may have negative values), the optimization algorithms to perform matrix factorization are usually faster than the inference algorithms for statistical models. In particular, the HEigen algorithm (Kang et al., 2011) computes the rank-k singular value decomposition (SVD) on networks with $N \geq 1$ billion nodes, given a cluster with hundreds of Hadoop machines. Furthermore, matrix factorization can be reinterpreted in a probabilistic setting, as seen in the work of Singh and Gordon (2010), who have developed a probabilistic matrix factorization framework for an arbitrary number of relational matrices. To incorporate network context into such a coupled matrix factorization framework, one would require at least two matrices: the adjacency matrix and one or more matrices of nodes against their features. Whether the resulting algorithm is scalable strongly depends on how the objective function is constructed. If the objective function is dense in the adjacency matrix (i.e., work is performed even on the zeros) such as in Wang et al. (2011), then the algorithm requires at least $\Omega(N^2)$ work. It will not finish in a realistic amount of time on networks with 100 million nodes, even if a large computer cluster is used. On the other hand, if the objective function is sparse (i.e., no work performed on the zeros), then one may take advantage of existing software for distributed, large-scale coupled matrix factorization, such as FlexiFaCT (Beutel et al., 2014). Another example of coupled matrix factorization is the linear-time PICS network analysis algorithm (Akoglu et al., 2012), which can analyze networks with $N \approx 75$k nodes in about 1-2 hours.

2. Data Representation: Triangular Motif Representation of Networks

In many real-world problems where computational cost is a bottleneck, transforming the original raw data into a task-dependent representation might well suffice for subsequent analyses and save significantly on computational cost. Classical examples include: (1) the *bag-of-words* representation of a document, in which the ordering information of words is discarded—although much grammatical information is lost, this representation has proven effective in natural language processing tasks such as topic modeling (Blei et al., 2003); (2) the use of *superpixels* to represent images, in which adjacent pixels are grouped into larger superpixels—the resulting image representation is compact, and leads to faster and better-performing segmentation algorithms in computer vision (Cao and Fei-Fei, 2007; Fulkerson et al., 2009).

Similarly, in probabilistic modeling of networks, the traditional dense adjacency matrix representation (Figure 1) is not suitable for the massive scales ($N \geq 100$ million) considered in this work. In particular, many probabilistic network algorithms touch every entry of



	1	2	3	4	5
1	-	1	1	0	0
2	1	-	0	1	1
3	1	0	-	1	0
4	0	1	1	-	1
5	0	1	0	1	-

Figure 1: Three different representations of the same network: as a graph, as an adjacency matrix, and as a list of 2-edge and 3-edge triangular motifs.

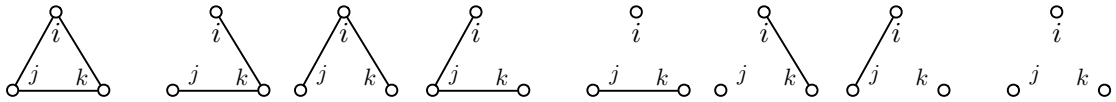


Figure 2: Four types of triangular motifs: (a) 3-edge triangle; (b) 2-edge triangle; (c) 1-edge triangle; (d) empty-triangle. For latent space inference of Internet-scale networks, we only focus on 3-edge and 2-edge triangles.

the adjacency matrix, leading to $\Theta(N^2)$ run-time complexity per iteration. To solve the representational and computational challenges of large-scale probabilistic network inference, we advocate the use of triangular motifs (Figures 1 and 2) as a foundation to achieve succinct yet informative representation of networks. Triangular motifs have historically played an important role in biological network analysis (Milo et al., 2002) as well as in social network analysis (Simmel and Wolff, 1950; Granovetter, 1973; Krackhardt and Handcock, 2007). After describing triangular motifs, we will show in the next few sections how they can be used to construct a mixed-membership network model with $O(CNK)$ per-iteration inference cost for a small constant C.

Given an *undirected* network[2] with N vertices, we shall use the term "1-edge" to refer to edges that exist between two vertices, and the term "0-edge" to refer to missing edges. A triangular motif E_{ijk} over 3 vertices $i < j < k$ is simply the type of subgraph exhibited by those 3 vertices. There are four basic classes of triangular motifs (Figure 2), distinguished by their number of 1-edges: 3-edge triangle Δ_3 (three 1-edges), 2-edge triangle Δ_2 (two 1-edges), 1-edge triangle Δ_1 (one 1-edge), and empty-triangle Δ_0 (no 1-edges). The total number of triangular motifs, over all four classes, is $\Theta(N^3)$. However, our goal is not to account for all four classes in a network representation; instead, we will focus on 3-edge and 2-edge triangles (Δ_3 and Δ_2) while ignoring 1-edge triangles and empty-triangles (Δ_1 and Δ_0). There are two key motivations for this approach:

2. Our approach can also be generalized to directed networks, though the analysis is more involved since directed networks can have more triangular motifs than undirected networks.

1. In the network literature, many important network measures and characteristics are quantified and captured by the three-node *connected* subgraphs (namely Δ_3 and Δ_2). To name a few: (1) the network clustering coefficient (Watts and Strogatz, 1998; Newman and Park, 2003), a common measure of triadic closure in social network theory (Simmel and Wolff, 1950; Granovetter, 1973; Krackhardt and Handcock, 2007), is defined as the relative number of 3-edge triangles compared to the total number of connected vertex triplets (i.e., 3-edge and 2-edge triangles); (2) the most significant and recurrent structural patterns in many complex networks, so-called "network motifs", are the three-node connected subgraphs (Milo et al., 2002). Therefore, one naturally expects the classes of 3-edge and 2-edge triangles to capture most of the informative structure in a large network.

2. The four classes of triangular motifs certainly contain redundant information, because in principle one needs only $\Theta(N^2)$ bits to fully describe a network. Though node triplets with only one or zero 1-edge (namely Δ_1 and Δ_0) are not uncommon in real-world networks (Leskovec et al., 2009), those information might already be contained in the list of Δ_3 and Δ_2 triangles. For instance, one can easily conclude from the 2/3-edge triangular motif representation of the network in Figure 1 that a 1-edge triangle Δ_1 is formed among the node triplet $\{1, 3, 5\}$. In fact, almost all network information found in an adjacency matrix representation is preserved by Δ_3 and Δ_2 triangles: every 1-edge that appears in some Δ_3 or Δ_2 can be identified in the corresponding triangular motif. The only exception is a set of isolated 1-edges. However, these small strongly connected components of size 2 can be easily detected and are less interesting from a large-scale community detection perspective.

One main advantage of characterizing a network in terms of triangular motifs Δ_3 and Δ_2 is that such representation is typically more compact than the adjacency matrix representation[3], via the following lemma:

Lemma 1 *The total number of Δ_3's and Δ_2's is $\Theta(\sum_i D_i^2)$, where D_i is vertex i's degree.*

Proof Let $\mathcal{N}_i$ be the neighbor set of vertex i. For each vertex i, form the set $\mathcal{T}_i$ of tuples (i, j, k) where $j < k$ and $j, k \in \mathcal{N}_i$, which represents the set of all pairs of neighbors of i. Because j and k are neighbors of i, the triangular motif formed among every tuple $(i, j, k) \in \mathcal{T}_i$ is either a 3-edge triangle Δ_3 or a 2-edge triangle Δ_2. It is easy to see that each Δ_2 is accounted for by exactly one $\mathcal{T}_i$, where i is the center vertex of the Δ_2, and that each Δ_3 is accounted for by three sets $\mathcal{T}_i, \mathcal{T}_j$ and $\mathcal{T}_k$, one for each vertex in the 3-edge triangle. Thus, $\sum_i |\mathcal{T}_i| = \sum_i \frac{1}{2}(D_i)(D_i - 1)$ is an upper bound to the total number of Δ_3's and Δ_2's. By modifying the preceding argument slightly, we can also show that the total number of Δ_3's and Δ_2's is bounded below by $\frac{1}{3}\sum_i |\mathcal{T}_i|$. ■

For networks with low maximum degree $D_{\max}$, $\Theta(\sum_i D_i^2) = O(N D_{\max}^2)$ is typically much smaller than $\Theta(N^2)$. Even in real-world networks with power-law behavior, the number of Δ_3's and Δ_2's tends to be still much smaller than the size of adjacency matrix N^2

3. Recall the size of data (including both 1-edges and 0-edges) in the adjacency matrix representation of a network is $\Theta(N^2)$, even if the data can be stored in a sparse manner.

Network	# nodes N	# edges M	adj matrix size N^2	Upper bound of $\#(\Delta_3, \Delta_2)$
DBLP	317,080	1,049,866	1.01×10^{11}	2.2×10^7
Amazon	334,863	925,872	1.12×10^{11}	9.8×10^6
Youtube	1,134,890	2,987,624	1.29×10^{12}	1.5×10^9
Livejournal	3,997,962	34,681,189	1.60×10^{13}	4.3×10^9
WDC Subdomain/Host	101 million	2 billion	1.02×10^{16}	1.0×10^{13}

Table 1: Networks used in this paper: number of 2-edge and 3-edge triangles versus adjacency matrix size. Many (though not all) probabilistic network models use the entire non-sparse adjacency matrix as input. The 2/3-edge triangular representation provides a more compact alternative—in all example networks, the upper bound for the number of Δ_3's and Δ_2's is at least 1000 times smaller than the full adjacency matrix.

(Table 1), and this allows us to construct a more efficient inference algorithm scalable to larger networks. However, the large number of triangles still poses storage and computational challenges. Our solution is to keep the triangular motifs in an *implicit* representation, and subsample a mini-batch of them only when they are needed (and discard them afterwards) in the stochastic variational inference algorithm. We will describe such technique in Section 5.1, and our results will surprisingly show that the inference algorithm only needs to touch a small fraction of triangles before converging to an accurate result.

3. Model: Scalable Network Model (STM) Based on Triangular Motifs

Given a network, now represented by a bag of triangular motifs Δ_3 and Δ_2 (Figure 1), the goal is to perform probabilistic inference of mixed-membership vectors of individual nodes, which can then be used for overlapping community detection. This is as opposed to traditional single-membership community detection, which assigns each vertex to exactly one community. By taking a mixed-membership approach, one gains many benefits over single-membership models, such as outlier detection, improved visualization, and better interpretability (Blei et al., 2003; Airoldi et al., 2008).

To construct the mixed-membership network model based on the triangular motif representation, which we call STM for scalable triangle model, we first establish some notation used throughout the paper. Recall that we are concerned only with 3-edge and 2-edge triangles (Δ_3 and Δ_2) in the triangular motif representation of networks. For each triplet of vertices $i, j, k \in \{1, \ldots, N\}, i < j < k$, if the subgraph on i, j, k is a 2-edge triangle with i, j, or k at the center, then let $E_{ijk} = 1$, 2, or 3 respectively; if a 3-edge triangle is formed among i, j, k, then let $E_{ijk} = 4$. In other words, E_{ijk} denotes the observed type of triangular motif on vertices i, j, k. Whenever the subgraph on i, j, k is a 1-edge triangle or an empty-triangle, we simply discard it (i.e., E_{ijk} is not part of the model) for the reasons that have been described in Section 2 .

Next, we assume K latent communities, and that each vertex takes a mixture distribution over them. The observed 2-edge and 3-edge triangles $\{E_{ijk}\}$ are generated according to a latent space model that defines (1) the mixture distribution over community memberships at each vertex, and (2) a tensor-like data structure of triangle-generating probabilities. More formally, each vertex i is associated with a community mixed-membership vector $\theta_i \in \Delta^{K-1}$ restricted to the $(K-1)$-simplex Δ^{K-1}. This mixed-membership vector θ_i is used to gen-

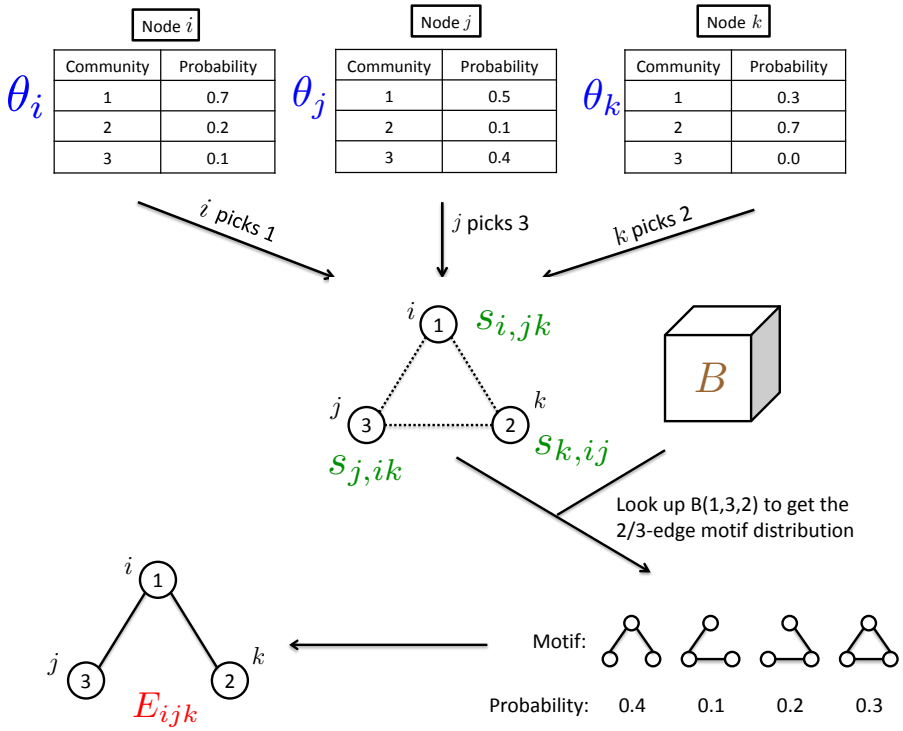


Figure 3: High-level generative process of how the mixed-membership vectors $\theta_i, \theta_j, \theta_k$ and a "tensor" of triangle-generating probabilities $\mathbf{B}$ are used to generate the triangular motif E_{ijk}. In this example, $(s_{i,jk} = 1, s_{j,ik} = 3, s_{k,ij} = 2)$ represents a context-dependent instantiation of communities for three vertices i, j, k when they are interacting, and the entry B_{132} is examined to obtain the multinomial parameter of generating E_{ijk}. All these probabilities are unobserved and need to be inferred from the observed triangular motifs.

erate community indicators $s_{i,jk} \in \{1, \ldots, K\}$, each of which represents the community chosen by vertex i when it is forming a triangle with vertices j and k. The probability of observing a triangular motif E_{ijk} depends on the community triplet $(s_{i,jk}, s_{j,ik}, s_{k,ij})$ and a tensor of triangle-generating parameters $\mathbf{B}$. Each element B_{xyz} of this tensor contains the multinomial probabilities of generating the four possible 2-edge and 3-edge triangles, when three vertices with respective communities x, y, z interact. A schematic of this generative process is illustrated in Figure 3.

3.1 Parsimonious Parameter Structure

The triangle-generating probability tensor $\mathbf{B}$ described above contains $O(K^3)$ distributions, one for each possible community combination (x, y, z). This presents challenging issues in both statistical estimation and computational complexity: (1) the large number of $O(K^3)$ parameters to be estimated leads to a loss of statistical efficiency (particularly when $K \geq 1000$); (2) inference requires $O(K^3)$ time per iteration, which is computationally intractable. An elegant solution to this problem is to reduce the number of triangle-generating

parameters by partitioning the $O(K^3)$ community combination space intro several groups, and then sharing parameters within the same group (Yin et al., 2013). Our parameter-sharing strategy is based on the number of distinct states in the configuration of the community triplet (x, y, z):

1. If all three communities are the same x, the triangle-generating probability is determined by B_{xxx} (this is identical to the diagonal of the $O(K^3)$ parameterization of $\mathbf{B}$ tensor). There are K such parameters $B_{111}, \ldots, B_{KKK}$.

2. If only two communities indices exhibit the same state x (called the majority community), the probability of triangles is governed by B_{xx}, no matter what the third, minority community y is. The intuition is that the identity of the majority community is the dominant factor in determining the triangle-generating probabilities, regardless of the minority community. As with any statistical assumption, this is never perfectly true for real data, but nevertheless turns out to be a good assumption, as our results will show. There are K such parameters $B_{11}, \ldots, B_{KK}$.

3. If the three community indices are all distinct, the probability of triangular motifs depends on B_0, a single parameter independent of the community identities. This is similar in spirit to the combined off-diagonal blockmatrix parameters used in hierarchical blockmodels (Ho et al., 2012a) and the assortative MMSB (Gopalan et al., 2012; Gopalan and Blei, 2013).

This sharing strategy yields just $O(K)$ parameters $B_0, B_{xx}, B_{xxx}, x \in \{1, \ldots, K\}$ (since there are at most four distinct 2-edge and 3-edge triangles), allowing STM to scale to far more communities than a naive $O(K^3)$ parameterization of $\mathbf{B}$ tensor.

3.2 Equivalence Classes of Triangles

For certain configurations of community triplet $(s_{i,jk}, s_{j,ik}, s_{k,ij})$, some of the triangular motifs can become indistinguishable. To illustrate, consider the example illustrated in Figure 4: suppose that nodes i and j take membership in community x (the majority community), while the node k takes community y—that is, we have $x = s_{i,jk} = s_{j,ik} \neq s_{k,ij} = y$. Under these assignments, one cannot distinguish the 2-edge triangle with i in the center (right panel, $E_{ijk} = 1$) from the triangle with j in the center (left panel, $E_{ijk} = 2$). This is because both are 2-edge triangles centered at a vertex with majority community x, and are thus *equivalent* with respect to the underlying community configuration $x - x - y$. Formally, this $x - x - y$ community configuration induces a set of triangle equivalence classes $\{\{1, 2\}, \{3\}, \{4\}\}$ of all possible triangular motifs $E_{ijk} \in \{1, 2, 3, 4\}$. We treat the triangular motifs within the same equivalence class as *stochastically equivalent*; that is, the conditional probabilities of events $E_{ijk} = 1$ and $E_{ijk} = 2$ are assumed to be the same if $x = s_{i,jk} = s_{j,ik} \neq s_{k,ij}$. All possible cases are enumerated as follows (see also Table 2):

1. If all three vertices have the same community x, all three 2-edge triangles are equivalent and the induced set of equivalence classes is $\{\{1, 2, 3\}, \{4\}\}$. The probability of E_{ijk} is determined by $B_{xxx} \in \Delta^1$, where $B_{xxx,1}$ represents the *total* probability of sampling an 2-edge triangle from $\{1, 2, 3\}$ and $B_{xxx,2}$ represents the 3-edge triangle probability. Thus, the probability of a particular 2-edge triangle is $B_{xxx,1}/3$.

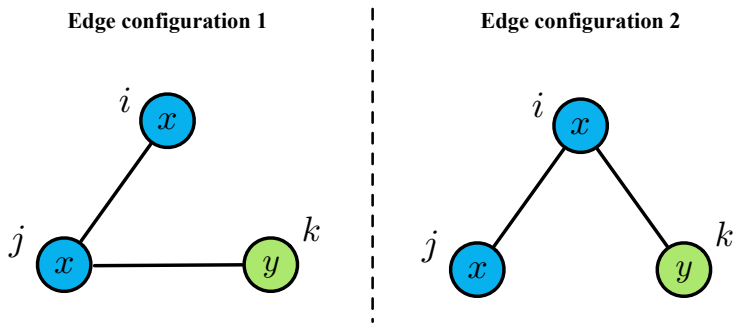


Figure 4: An example of two indistinguishable, equivalent motifs. Although the edge configuration is different in two motifs ($i - j - k$ on the left versus $j - i - k$ on the right), the motifs are indistinguishable under the given community combination, because they are both cases in which the central node (j and i respectively) has the majority community x.

2. If only two vertices have the same community x (majority community), the probability of E_{ijk} is governed by $B_{xx} \in \Delta^2$. Here, $B_{xx,1}$ and $B_{xx,2}$ represent the 2-edge triangle probabilities (for 2-edge triangles centered at a vertex in majority and minority community respectively), and $B_{xx,3}$ represents the 3-edge triangle probability. There are two possible 2-edge triangles with a vertex in majority community at the center, and hence each has probability $B_{xx,1}/2$.

3. If all three vertices have distinct community, the probability of E_{ijk} depends on $B_0 \in \Delta^1$, where $B_{0,1}$ represents the *total* probability of sampling an 2-edge triangle from $\{1, 2, 3\}$ (regardless of the center vertex's community) and $B_{0,2}$ represents the 3-edge triangle probability.

3.3 Generative Process for Scalable Triangular Model (STM)

We summarize the generative process for a bag of triangular motifs under the STM:

1. Draw $B_0 \in \Delta^1$, $B_{xx} \in \Delta^2$ and $B_{xxx} \in \Delta^1$ for each community $x \in \{1, \ldots, K\}$, according to symmetric Dirichlet distributions Dirichlet(λ).

2. For each vertex $i \in \{1, \ldots, N\}$, draw a mixed-membership vector $\theta_i \sim \text{Dirichlet}\,(\alpha)$.

3. For each triplet of vertices $(i, j, k)\,, i < j < k$,

   (a) Draw community configuration $s_{i,jk} \sim \text{Discrete}\,(\theta_i)$, $s_{j,ik} \sim \text{Discrete}\,(\theta_j)$, $s_{k,ij} \sim \text{Discrete}\,(\theta_k)$.

   (b) Draw an *equivalence class* of triangular motifs $\{E_{ijk}\}$ based on B_0, B_{xx}, B_{xxx} and the configuration of $(s_{i,jk}, s_{j,ik}, s_{k,ij})$. See Table 2 for the conditional probabilities of each equivalence class.

   (c) Draw a triangular motif E_{ijk} uniformly at random from the chosen equivalence class (this is what we meant by *stochastic equivalence* earlier).

$(s_{i,jk}, s_{j,ik}, s_{k,ij})$	Equivalence classes	Conditional probability of $E_{ijk} \in \{1,2,3,4\}$
$x = s_{i,jk} = s_{j,ik} = s_{k,ij}$	$\{1,2,3\},\{4\}$	$\text{Discrete}\big(\big[\frac{B_{xxx,1}}{3}, \frac{B_{xxx,1}}{3}, \frac{B_{xxx,1}}{3}, B_{xxx,2}\big]\big)$
$x = s_{i,jk} = s_{j,ik} \neq s_{k,ij}$	$\{1,2\},\{3\},\{4\}$	$\text{Discrete}\big(\big[\frac{B_{xx,1}}{2}, \frac{B_{xx,1}}{2}, B_{xx,2}, B_{xx,3}\big]\big)$
$x = s_{i,jk} = s_{k,ij} \neq s_{j,ik}$	$\{1,3\},\{2\},\{4\}$	$\text{Discrete}\big(\big[\frac{B_{xx,1}}{2}, B_{xx,2}, \frac{B_{xx,1}}{2}, B_{xx,3}\big]\big)$
$x = s_{j,ik} = s_{k,ij} \neq s_{i,jk}$	$\{2,3\},\{1\},\{4\}$	$\text{Discrete}\big(\big[B_{xx,2}, \frac{B_{xx,1}}{2}, \frac{B_{xx,1}}{2}, B_{xx,3}\big]\big)$
$s_{k,ij} \neq s_{i,jk} \neq s_{j,ik}$	$\{1,2,3\},\{4\}$	$\text{Discrete}\big(\big[\frac{B_{0,1}}{3}, \frac{B_{0,1}}{3}, \frac{B_{0,1}}{3}, B_{0,2}\big]\big)$

Table 2: Equivalence classes and conditional probabilities of E_{ijk} given $s_{i,jk}, s_{j,ik}, s_{k,ij}$ (see text for details).

It is worth noting that STM is not a generative model of networks since (a) empty-triangles and 1-edge triangles are not modeled, and (b) one can possibly generate a set of triangles that does not correspond to any network. This is a consequence of using a bag-of-triangles model in which the generative process does not force overlapping triangles to have consistent edge values. For example, generating a 2-edge triangle centered at i for (i,j,k) followed by a 3-edge triangle for (j,k,ℓ) can produce a mismatch on the edge (j,k). However, given a bag of triangular motifs $\mathbf{E}$ extracted from a network, the above procedure defines a valid probabilistic model $p(\mathbf{E} \mid \alpha, \lambda)$, and we can use it for performing posterior inference $p(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B} \mid \mathbf{E}, \alpha, \lambda)$. We stress that our goal is fast and scalable latent space inference for overlapping community detection, not network simulation[4].

4. Inference: Efficient Stochastic Variational Inference (SVI) for STM

In this section, we present a stochastic variational inference (SVI) algorithm (Hoffman et al., 2013) with $O(CNK)$ per-iteration cost for performing approximate inference under our STM (where C is a small constant). The inferred posterior probability distribution of the latent variables, in particular the mixed-membership vectors θ_i, can then be used for overlapping community discovery. The high-level ideas are to (1) develop a structured mean-field approximation—a more accurate approximation to the true posterior than the naive mean-field solution explored in earlier mixed-membership network models (Airoldi et al., 2008), but requires $O(K^3)$ run-time per triangular motif; (2) refine the structured mean-field approximation to lower the run-time complexity to $O(K)$ while maintaining its high accuracy, inspired by a careful understanding of the typical structure of the posterior distribution; (3) apply stochastic variational inference (SVI) to yield a faster approximate inference algorithm; (4) propose a new stochastic subsampling strategy in the SVI algorithm that empirically achieves high-quality results without having to touch every 2/3-edge triangle in the network, thus ensuring scalability to very large networks which may have more than trillions (10^{12}) of triangles.

4. For applications in which simulation is vital, it is possible to design a variant of STM that sequentially generates motifs so as to have consistent edge values across overlapping triangles. In such model, all possible triples (i,j,k) are sequentially considered, and a new motif among them is simulated conditioning on the patterns of other motifs that have already been generated. This non-exchangeable model is, however, out of the scope of this work.

4.1 Structured Mean-field Approximation

As with other mixed-membership models, computing the true posterior of the latent variables $p(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B} \mid \mathbf{E}, \alpha, \lambda)$ under the STM is intractable, and approximate inference must be employed. There are three common strategies to perform approximate inference in mixed-membership models: Markov chain Monte Carlo (MCMC) (Griffiths and Steyvers, 2004), variational inference (Blei et al., 2003; Airoldi et al., 2008), and spectral methods (Anandkumar et al., 2014). MCMC methods are typically viewed as the "gold standard" as they are guaranteed to eventually (but could be slow in practice) converge to the true posterior. Variational inference provides a fast approximation that can be accurate enough for the task at hand, provided care has been taken to design an appropriate factorized variational distribution. Spectral methods have shown much promise in terms of accuracy and scalability, but the $O(K^3)$ run-time complexity limits its application to the setting of $K \geq 1000$ we are considering.

Based on these considerations, we choose a structured mean-field approximation for the STM model. The corresponding inference algorithm proceeds via coordinate ascent update on an objective function known as "variational lower bound" that depends on a set of "variational parameters". To construct the variational lower bound, we begin by approximating the true (but intractable) posterior $p(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B} \mid \mathbf{E}, \alpha, \lambda)$ by a *partially* factorized distribution $q(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B})$,

$$
\begin{aligned}
q(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B}) &= q(\mathbf{s} \mid \boldsymbol{\phi}) q(\boldsymbol{\theta} \mid \boldsymbol{\gamma}) q(\mathbf{B} \mid \boldsymbol{\eta}) \\
&= \left[\prod_{(i,j,k) \in I} q(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \phi_{ijk}) \right] \left[\prod_{i=1}^{N} q(\theta_i \mid \gamma_i) \right] \left[\prod_{x=1}^{K} q(B_{xxx} \mid \eta_{xxx}) \right] \left[\prod_{x=1}^{K} q(B_{xx} \mid \eta_{xx}) \right] [q(B_0 \mid \eta_0)],
\end{aligned} \tag{1}
$$

where I is the set of all 2/3-edge triangles in the network, i.e., $I = \{(i,j,k) : i < j < k, E_{ijk} \in \{1,2,3,4\}\}$. The factorized distribution $q(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B})$ contains 3 types of terms:

1. Terms for the community triplet distribution for each triangle, $q(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \phi_{ijk})$;
2. Terms for the mixed-membership community distribution for each node, $q(\theta_i \mid \gamma_i)$;
3. Terms for the triangle-generating distributions, $q(B_{xxx} \mid \eta_{xxx})$, $q(B_{xx} \mid \eta_{xx})$, and $q(B_0 \mid \eta_0)$.

In particular, we have defined a *joint* (as opposed to fully factorized) distribution $q(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \phi_{ijk})$ over the community triplet $(s_{i,jk}, s_{j,ik}, s_{k,ij})$:

$$
q(s_{i,jk} = x, s_{j,ik} = y, s_{k,ij} = z) \doteq q_{ijk}(x,y,z) = \phi_{ijk}^{xyz}, \quad x,y,z = 1, \ldots, K. \tag{2}
$$

The posterior $q(\theta_i)$ is a Dirichlet(γ_i), and the posteriors of B_{xxx}, B_{xx}, B_0 are parameterized as: $q(B_{xxx}) = \text{Dirichlet}(\eta_{xxx}), q(B_{xx}) = \text{Dirichlet}(\eta_{xx})$, and $q(B_0) = \text{Dirichlet}(\eta_0)$.

Rationale for the Structured Approximation. Conditioning on the observed type of triangular motif E_{ijk} on vertex triplet (i,j,k)—either a 2-edge triangle Δ_2 or a 3-edge triangle Δ_3, the community indicators $(s_{i,jk},\ s_{j,ik},\ s_{k,ij})$ are often highly correlated. For example, suppose that nodes i,j,k form a 3-edge triangle, then it is empirically and sociologically far more likely that they all belong to the same community, as opposed to being a

mix of different communities. As a result, the true posterior distribution $p(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \boldsymbol{\theta}, \mathbf{B}, \mathbf{E}, \alpha, \lambda)$—a discrete distribution over K^3 events—tend to concentrate around (some of) the K "all-the-same-community" events $\{s_{i,jk} = s_{j,ik} = s_{k,ij} = 1, 2, \ldots, K\}$. If we think of the posterior as a third-order tensor of dimension $K \times K \times K$ in which each element (x, y, z) represents the posterior probability $p(s_{i,jk} = x, s_{j,ik} = y, s_{k,ij} = z \mid \boldsymbol{\theta}, \mathbf{B}, \mathbf{E}, \alpha, \lambda)$, we see that it is very likely to be a tensor with rank greater than one due to its non-zero diagonal entries. Therefore, it cannot be accurately approximated by a fully factorized distribution $q(s_{i,jk} \mid \phi_{i,jk})q(s_{j,ik} \mid \phi_{j,ik})q(s_{k,ij} \mid \phi_{k,ij})$ as used in the naive mean-field approximation, because the outer product yields a rank-1 tensor. A similar argument holds for 2-edge triangles, whose nodes are likely to be in one or two communities, with a similar correlation structure as to the 3-edge case. For this reason, we have chosen a joint form $q(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \phi_{ijk})$ for the variational posterior, in order to capture the correlation structure of the true posterior $p(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \boldsymbol{\theta}, \mathbf{B}, \mathbf{E}, \alpha, \lambda)$.

However, having a structured approximation is computationally more expensive: the variational parameter ϕ_{ijk} used in the variational posterior $q(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \phi_{ijk})$ is a tensor containing K^3 entries, which would cause an unacceptable $O(K^3)$ run-time per triangular motif if handled naively. In the next section, we will discuss a more refined approximation strategy that maintains the high accuracy of the structured mean-field approximation but only requires $O(K)$ run-time per triangular motif, thus ensuring scalability to thousands of communities K.

4.2 Stochastic Variational Inference (SVI) Algorithm

The structured mean-field approximation (Wainwright and Jordan, 2008) aims to minimize the KL divergence $\mathrm{KL}(q \parallel p)$ between the approximating distribution q and the true posterior p; it is equivalent to maximizing a lower bound $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma})$ of the log marginal likelihood of the triangular motifs (based on Jensen's inequality) with respect to the variational parameters $\{\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma}\}$ defined in Equation 1:

$$\log p(\mathbf{E} \mid \alpha, \lambda) \geq \mathbb{E}_q[\log p(\mathbf{E}, \mathbf{s}, \boldsymbol{\theta}, \mathbf{B} \mid \alpha, \lambda)] - \mathbb{E}_q[\log q(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B})] \doteq \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma}). \tag{3}$$

To simplify the notation, we decompose the variational objective $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma})$ into a "global" term and a summation of "local" terms, where each local term corresponds to one 2/3-edge triangle in the network (see Appendix A.1 for their exact forms).

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma}) = g(\boldsymbol{\eta}, \boldsymbol{\gamma}) + \sum_{(i,j,k) \in I} \ell(\phi_{ijk}, \boldsymbol{\eta}, \boldsymbol{\gamma}). \tag{4}$$

The global term $g(\boldsymbol{\eta}, \boldsymbol{\gamma})$ depends only on the global variational parameters $\boldsymbol{\eta}$, which govern the posterior of the triangle-generating probabilities $\mathbf{B}$, as well as the per-node mixed-membership parameters $\boldsymbol{\gamma}$. Each local term $\ell(\phi_{ijk}, \boldsymbol{\eta}, \boldsymbol{\gamma})$ depends on per-triangle variational parameter ϕ_{ijk} as well as the global parameters. Our distributed and parallel implementation strategy will involve splitting the inferential work along disjoint sets of local variational parameters, as we will show in Algorithm 1.

To understand how an SVI algorithm accelerates approximate inference, let us consider Equation 4 with the local parameters $\boldsymbol{\phi}$ being maximized out. Define $\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma}) \doteq$

$\max_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma})$, i.e., the variational objective achieved by fixing the global parameters $\boldsymbol{\eta}, \boldsymbol{\gamma}$ and optimizing the local parameters $\boldsymbol{\phi}$:

$$\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma}) = g(\boldsymbol{\eta}, \boldsymbol{\gamma}) + \sum_{(i,j,k)\in I} \max_{\phi_{ijk}} \ell(\phi_{ijk}, \boldsymbol{\eta}, \boldsymbol{\gamma}). \tag{5}$$

The idea behind SVI (Hoffman et al., 2013) is as follows: instead of computing the gradient $\nabla\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$ to perform the variational inference of the global parameters $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$, which involves a costly summation over all triangular motifs as in Equation 5, an unbiased noisy approximation of the gradient can be obtained much more cheaply by summing over a small subsample of triangles. One can then use this approximate gradient in the stochastic gradient ascent algorithm (Bottou, 2004) to maximize the variational objective $\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$. With this unbiased estimate of the gradient and a suitable adaptive step size, the algorithm is guaranteed to converge (in probability) to a stationary point of the variational objective $\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$ (Robbins and Monro, 1951). It should be noted that the subsampling strategy comes at the cost of introducing a variance in gradient computation, and hence more iterations might be required for convergence. However, the time taken by each iteration can be greatly reduced to a small fraction of the original time; for example, if we subsample 10% of the triangles, then only 10% computation time is required to obtain the approximate gradient. In fact, there is significant empirical evidence that SVI algorithm can lead to faster convergence on various problems (Hoffman et al., 2013; Yin et al., 2013).

Subsampling Strategy. In our setting, the most natural way to obtain an unbiased gradient of $\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$ is to sample a "mini-batch" of triangular motifs at each iteration, and then average the gradient of local terms in Equation 5 *only* for these sampled triangles. Formally, let $m = |I|$ be the total number of 2/3-edge triangles[5] and define

$$\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma}) = g(\boldsymbol{\eta}, \boldsymbol{\gamma}) + \frac{m}{|S|} \sum_{(i,j,k)\in S} \max_{\phi_{ijk}} \ell(\phi_{ijk}, \boldsymbol{\eta}, \boldsymbol{\gamma}), \tag{6}$$

where S is a mini-batch of triangles sampled uniformly at random. It is easy to verify that $\mathbb{E}_S[\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})] = \mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$, hence $\nabla\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$ is unbiased: $\mathbb{E}_S[\nabla\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})] = \nabla\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$.

In our implementation, we subsample C pairs of neighbors for each node $i \in \{1, \ldots, N\}$ in parallel, where C is a small constant[6], resulting in a mini-batch set S containing CN triangles in each SVI iteration. Such fast parallel subsampling procedure, however, may induce a bias in the estimate of the gradient because the CN triangles in S are not necessarily sampled from the space of all triangles uniformly at random. We defer the discussion of this subtlety to Section 5.1, after the detailed description of our implementation. The performance on overlapping community detection in Section 6 shows that our SVI algorithm converges to a high-quality result without having to touch all 2/3-edge triangles in the network, i.e., $CNt_{\max} \ll m$, where $t_{\max}$ is the number of iterations until convergence.

5. The total number of 2/3-edge triangles in large networks may be computationally infeasible to count—in fact, triangle counting is a hard problem in its own right and an area of active research (Low et al., 2012). In Lemma 1, we have shown that $\frac{1}{3}T \le m \le T$ where $T = \sum_i \frac{1}{2}(D_i)(D_i - 1)$. In our implementation, we simply let $m \approx T$, thus approximating m to within a factor of 3 of its true value.
6. All our experiments are conducted with the minimum value $C = 1$, which is shown to be sufficiently accurate for large-scale networks (Section 6.4).

Exact Local Update. To obtain the gradient $\nabla\mathcal{L}_S(\boldsymbol{\eta},\boldsymbol{\gamma})$, one needs to compute the optimal local variational parameters ϕ_{ijk} (keeping $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ fixed) for each sampled triangle (i,j,k) in the mini-batch S; these optimal ϕ_{ijk}'s are then used in Equation 6 to compute $\nabla\mathcal{L}_S(\boldsymbol{\eta},\boldsymbol{\gamma})$. Taking partial derivatives of Equation 4 with respect to each local term ϕ_{ijk}^{xyz} and setting them to zero, we get for distinct $x,y,z\in\{1,\ldots,K\}$,

$$\phi_{ijk}^{xyz}\propto\exp\Big\{\mathbb{E}_q[\log B_{0,2}]\mathbb{I}[E_{ijk}=4]+\mathbb{E}_q[\log(B_{0,1}/3)]\mathbb{I}[E_{ijk}\neq 4]+\mathbb{E}_q[\log\theta_{i,x}+\log\theta_{j,y}+\log\theta_{k,z}]\Big\}.$$

See Appendix A.2 for the update equations of ϕ_{ijk}^{xxx} and ϕ_{ijk}^{xxy} $(x\neq y)$.

O(K) **Approximation to Local Update.** As explained earlier, because there are K^3 variational parameters ϕ_{ijk}^{xyz} for each sampled triangle (i,j,k), the exact local update requires $O(K^3)$ work to solve for all ϕ_{ijk}^{xyz}, making it unscalable. We now describe an $O(K)$ approximation that is almost as accurate as the exact $O(K^3)$ update (and is certainly far more accurate than a fully factorized approximation as in the naive mean-field approach). To enable a faster local update, we replace $q_{ijk}(x,y,z\mid\phi_{ijk})$ in Equation 2 with a simpler, but still structured (as opposed to fully factorized), "mixture-of-deltas" variational distribution,

$$q_{ijk}(x,y,z\mid\delta_{ijk})=\sum_{(a,b,c)\in\mathcal{A}}\delta_{ijk}^{abc}\,\mathbb{I}[x=a,y=b,z=c],$$

where $\mathcal{A}$ is a set containing O(K) strategically chosen triples (a,b,c), and $\sum_{(a,b,c)\in\mathcal{A}}\delta_{ijk}^{abc}=1$. In other words, this "mixture-of-deltas" variational distribution is still a discrete probability distribution over K^3 events, but its probability mass is assumed to fall entirely on the O(K) randomly chosen entries and community combinations not in $\mathcal{A}$ are assumed to have zero probability. Conveniently, the update equations for the O(K) free parameters δ_{ijk}^{abc} are practically identical to the full $O(K^3)$ update equations for ϕ_{ijk} (Equations 9, 10 and 11 in Appendix A.2). The only difference is normalization, which is now performed over the set of O(K) δ's instead of all K^3 events. Since we subsample CN triangles in each SVI iteration, the total time complexity of local update is $O(CNK)$ per iteration.

Careful selection of the O(K) entries $(a,b,c)\in\mathcal{A}$ is critical to the performance of the O(K) "mixture-of-deltas" approximation. In our implementation, we choose these entries in $\mathcal{A}$ based on the observation that most nodes in real networks belong to just a few communities (Yang and Leskovec, 2012b), and hence the true mixed-membership vectors are also concentrated around just a few entries. In fact, more than 90% of nodes in most real-world networks studied in this paper have no more than 10 ground-truth communities. This suggests that for any triangle (i,j,k), the number of plausible assignments to its community triplet $(s_{i,jk},s_{j,ik},s_{k,ij})$ is usually very small compared to K^3, and we should focus our choice on triplets (a,b,c) such that the variational posteriors $q(\theta_{i,a}),q(\theta_{j,b}),q(\theta_{k,c})$ are all large. Furthermore, it is reasonable to add $O(K)$ random community combinations into consideration so as to enable the SVI algorithm to explore other communities to avoid becoming stuck in a local optimum. We employ the following procedure to construct $\mathcal{A}$:

`STM-ChooseA:`

1. Find the U_0 largest communities in each of $q(\theta_i)$, $q(\theta_j)$ and $q(\theta_k)$, and insert them into a set R_{core} ($|R_{\text{core}}|\leq 3U_0$). In our experiments, we use $U_0=10$ because empirically most nodes have no more than 10 communities.

2. Pick U_1 communities uniformly at random from $\{1, \ldots, K\}$, and put them into a set R_{random}. In our experiments, we use $U_1 = K/10$ so that $U_1 > U_0$ for large K, thus encouraging exploration.

3. Combine both sets: $R = R_{\text{core}} \cup R_{\text{random}}$. Note that $|R| = O(K)$.

4. Generate $|R|$ diagonal community combinations $\mathcal{A}_{\text{diag}} = \{(a, a, a) \text{ for all } a \in R\}$.

5. Generate $3|R|$ off-diagonal community combinations $\mathcal{A}_{\text{off}}$. Each combination is generated as follows: first draw $a \in R$ uniformly at random, then draw $b \in R$ where $b \neq a$. Finally, draw o uniformly at random from $\{1, 2, 3\}$. If $o = 1$, add (a, a, b); if $o = 2$, add (a, b, a); and if $o = 3$, add (b, a, a).

6. Generate $3|R|$ off-off-diagonal community combinations $\mathcal{A}_{\text{offoff}}$. Each combination is generated as follows: first draw $a \in R$ uniformly at random, then draw $b \in R$ where $b \neq a$, and finally draw $c \in R$ where $c \neq a, c \neq b$. Add (a, b, c) to $\mathcal{A}_{\text{offoff}}$.

7. Combine all three sets to obtain $\mathcal{A} = \mathcal{A}_{\text{diag}} \cup \mathcal{A}_{\text{off}} \cup \mathcal{A}_{\text{offoff}}$.

Note that we do not pick all possible community combinations for $\mathcal{A}_{\text{off}}$ and $\mathcal{A}_{\text{offoff}}$, which are of size $\mathrm{O}(K^2)$ and $\mathrm{O}(K^3)$, respectively. This is an intentional trade-off; we limit the size of $\mathcal{A}$ to $\mathrm{O}(K)$ to keep the inference feasible[7]. Because we re-select $\mathcal{A}$ every time we perform the local variational update for some triangle (i, j, k), the SVI algorithm still manages to explore the most likely community combinations with reasonably high probability, thus avoiding any bias due to a single choice of $\mathcal{A}$. In practice, we find that this $\mathrm{O}(K)$ "mixture-of-deltas" approximation works nearly as well as the full parameterization in Equation 2, while requiring only $O(K)$ work per sampled triangle. Finally, we stress that the "mixture-of-deltas" variational approximation is theoretically well-justified and not a heuristic. Standard variational inference theory states that any choice of $\mathcal{A}$ yields a valid lower bound to the log marginal likelihood at the current iteration, and therefore we are always updating variational parameters to maximize some variational lower bound.

Global Update. We appeal to stochastic natural gradient ascent (Amari, 1998; Sato, 2001; Hoffman et al., 2013) to optimize the global parameters $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$, as it greatly simplifies the update rules while maintaining the same asymptotic convergence properties as classical stochastic gradient. The natural gradient $\tilde{\nabla}\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$ is obtained by a premultiplication of the ordinary gradient $\nabla\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$ with the inverse of the Fisher information of the variational posterior q. See Appendix A.3 for the exact forms of the natural gradients with respect to $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$. To update the parameters $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$, we apply the stochastic natural gradient ascent rule

$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t + \rho_t \tilde{\nabla}_{\boldsymbol{\eta}} \mathcal{L}_S(\boldsymbol{\eta}_t, \boldsymbol{\gamma}_t), \quad \boldsymbol{\gamma}_{t+1} = \boldsymbol{\gamma}_t + \rho_t \tilde{\nabla}_{\boldsymbol{\gamma}} \mathcal{L}_S(\boldsymbol{\eta}_t, \boldsymbol{\gamma}_t), \tag{7}$$

7. A similar variational approximation was employed in our earlier development (Yin et al., 2013). The most salient difference is that the `STM-ChooseA` uses significantly fewer variational parameters (and is therefore faster), while still maintaining high experimental accuracy. In particular, for $K \geq 1000$, the choices of $U_0 = 10$ and $U_1 = K/10$ make the SVI algorithm almost 10 times faster than the method of Yin et al. (2013).

where the step size is given by $\rho_t = \tau_0(\tau_1 + t)^{-\kappa}$. To ensure convergence, the τ_0, τ_1, κ are set such that $\sum_t \rho_t^2 < \infty$ and $\sum_t \rho_t = \infty$. The specific values used in our experiments are $\tau_0 = 50$, $\tau_1 = 10000$, and $\kappa = 0.5$. The global update only costs $O(NK)$ time per iteration due to the parsimonious $O(K)$ parameterization of the triangle-generating probability tensor $\mathbf{B}$ (and hence its variational parameter $\boldsymbol{\eta}$) in our STM (Section 3.1).

Initialization. The SVI algorithm described above searches for a local maximum in a highly non-concave variational objective function (Equation 5). In principle, this makes it sensitive to the choice of initialization or seeding, a property shared by other overlapping community detection algorithms (Xie et al., 2013). If the initialization is not too far from the true community structure, then the algorithm will usually find the correct communities. However, a completely random initialization will often merge many neighboring communities into one giant community, which is undesirable.

To address this issue, we conduct the initialization of the SVI algorithm systematically, which is highly effective on all the ground-truth networks we test on (Section 6.1). It consists of two steps. First, we renumber all node indices via the following procedure:

`STM-CANONIZE`**:**

1. Initialize `MAP` to an empty dictionary and `COUNT` $= 1$. When the algorithm terminates, `MAP`$[i] = a$. In other words, we have renumbered the old node index i to the new node index a.

2. For each edge (i, j) in the edge list:

   (a) If i is not in `MAP`, then assign `MAP`$[i]$ = `COUNT` and `COUNT` = `COUNT` $+ 1$.

   (b) If j is not in `MAP`, then assign `MAP`$[j]$ = `COUNT` and `COUNT` = `COUNT` $+ 1$.

3. For each edge (i, j) in the edge list:

   (a) Output the re-indexed edge (`MAP`$[i]$, `MAP`$[j]$).

This procedure requires only linear-time work in the number of edges M, and executes in less than 10 minutes for all ground-truth networks in our experiments. In addition to renumbering all nodes in the range $[1, N]$, `STM-CANONIZE` often creates many sequences of contiguous node indices $a, a{+}1, a{+}2, \ldots, a{+}b$ such that a is connected to $a{+}1, a{+}2, \ldots, a{+}b$. The rationale is that nodes with numerically close indices are more likely to be close in terms of network distance, and thus are likely to be in the same community. This happens because the input edge list is not randomly ordered in practice, but rather usually groups adjacent edges together—as an example of how this may happen, when an adjacency matrix is converted to an edge list by scanning the rows one at a time, edges connecting to same node are adjacent in the edge list. When `STM-CANONIZE` encounters adjacent edges whose nodes have not been seen before, it gives those nodes contiguous indices.

The second step takes advantage of these contiguous sequences, by initializing contiguous blocks of node indices to different communities: the first N/K nodes are seeded to community 1, the second N/K nodes are seeded to community 2, and so on. Recall that γ_i's are the variational parameters for the mixed-membership vectors θ_i's; to seed node i to

community k, we initialize $\gamma_{i,k} = 10K$ and the remaining non-seeded elements of γ_i to be small, randomly-generated numbers close to 1.

For the variational parameters $\boldsymbol{\eta}$ corresponding to the triangle-generating probabilities $\mathbf{B}$, we initialize them as follows: $[\eta_{xxx,1}, \eta_{xxx,2}] = [1, 3]$, and $[\eta_{xx,1}, \eta_{xx,2}, \eta_{xx,3}] = [2, 1, 1]$ and $[\eta_{0,1}, \eta_{0,2}] = [3, 1]$. This reflects the intuition that three nodes with the same community are likely to form a 3-edge triangle, whereas for other cases (two nodes have the same community or three nodes have distinct community), it is likely to form a 2-edge triangle. Note that there is no need to initialize the local variational parameters $q_{ijk}(x, y, z)$, as they are solved through fixed-point iteration given the current values of $\boldsymbol{\gamma}, \boldsymbol{\eta}$ (Equations 9, 10 and 11). Finally, we fix the hyperparameters of $\boldsymbol{\theta}, \mathbf{B}$ to $\alpha = \lambda = 0.1$.

5. Distributed Implementation for Internet-Scale Networks

In order to apply our SVI algorithm to massive networks, we turn to its distributed implementation. The specific challenges we are facing include:

1. Big data: given that contemporary server machines only contain between 16GB to 256GB of memory, a massive network and its triangular representation cannot fit into the memory of a single machine. For example, the 101-million-node, 2-billion-edge web graph (Section 7) requires over 30GB to simply store it as an adjacency list, and the full triangular representation would be many orders of magnitude larger (in the TB range).

2. Big model: similarly, for a massive network, the STM parameters cannot fit into the memory of a single machine. With $N = 100$ million nodes and $K = 1000$ communities, we would need 400GB just to store the mixed-membership vectors θ_i's.

3. Slow inter-machine communication: typical network speeds range from 1Gbps to 10Gbps, hence inter-machine communication is several orders of magnitude slower than CPU-to-RAM communication. This means that we cannot synchronize parameters as frequently as in the single machine setting.

Unfortunately, the existing Hadoop MapReduce framework (Hadoop, 2012) is not well-suited to implement iterative convergent algorithms such as our SVI, because every map-reduce iteration incurs significant overheads and consequently takes orders of magnitude longer than other systems (Zaharia et al., 2010; Low et al., 2010). Another concern is that the map-reduce programming model does not provide a natural way to store model parameters in a persistent and distributed fashion, a challenge which has recently been addressed by high-performance parameter servers (Li et al., 2013; Ho et al., 2013). Based on these considerations, we develop our distributed-parallel SVI algorithm for STM (Algorithm 1) on top of the Petuum parameter server (Ho et al., 2013; Lee et al., 2014; Dai et al., 2015), a recent framework for general-purpose machine learning on big data and models. The high-level ideas to tackle challenges outlined above are to (1) keep the triangles in an implicit representation and use disk-based access to save machine memory; (2) partition the model over worker machines and exploit parameter sparsity to further reduce memory usage; (3) use a bounded-asynchronous communication technique to strike a balance between parameter accuracy and speed. We now discuss detailed approaches to each idea.

Algorithm 1 Distributed-parallel Stochastic Variational Inference for STM

1: $t = 0$. Initialize the global parameters $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$, and store them in the Petuum parameter server (`www.petuum.org`) for global access by any worker thread in the cluster.
2: Repeat the following steps in parallel until convergence. Parallelization is conducted in a data-parallel fashion: each worker thread in the cluster is responsible for a disjoint set of nodes, and all N nodes are collectively covered by all workers.

(1) Sample CN triangles to form a mini-batch S. We do this by sampling C pairs of neighbors for each node $i \in \{1, \ldots, N\}$ in parallel (see Section 5.1 for details of implementation).
(2) Optimize the local parameters $q_{ijk}(x, y, z)$ for all sampled triangles in parallel by Equations 9, 10 and 11.
(3) Accumulate sufficient statistics for the natural gradients of $\boldsymbol{\eta}, \boldsymbol{\gamma}$, and then discard local parameters $q_{ijk}(x, y, z)$ and the mini-batch S. Since the sufficient statistics are additive, we use the Petuum parameter server to accumulate them in parallel at each worker thread.
(4) Optimize the global parameters $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ by the stochastic natural gradient ascent rule (Equations 7), and then distribute the new global parameters to the worker threads.
(5) Update the step size for the next iteration: $\rho_t \leftarrow \tau_0(\tau_1 + t)^{-\kappa}$, $t \leftarrow t + 1$.

5.1 Handling Big Network Data and Trillions of Triangles

As shown in Lemma 1, the number of 2/3-edge triangles is bounded below by $\frac{1}{3}\sum_i \frac{1}{2}(D_i)(D_i - 1)$. This means even a single node with 1 million neighbors, which can occur in very large networks ($N \geq 100$ million nodes and $M \geq 1$ billion edges) with a power-law degree distribution, will contribute half a trillion triangles to the triangular representation and hence require TBs of storage.

Implicit Triangle Representation and Disk-based Storage. Our solution is to keep the triangles in an *implicit* representation, where we leverage the stochastic nature of the SVI algorithm and only sample triangles only when they are needed in step (1) of Algorithm 1 (and discard them afterwards). Specifically, we represent the original network using the following three data structures:

1. An adjacency list `Adj` represented as a dictionary (key-value) data structure. If the a-th neighbor of i is j, then the dictionary contains $\texttt{Adj}[(i, a)] = j$. While this key-value schema seems unorthodox compared to storing the entire neighbor set $\mathcal{N}_i$ as the value, they are easier to store at large scale and possibly can help improve CPU cache efficiency because values j are fixed-byte-size scalars whereas sets of neighbors $\mathcal{N}_i$ can have arbitrary byte length.

2. An edge list `Edg` represented as a set of edge tuples (i, j).

3. A degree list `Deg` represented as a vector, where $\texttt{Deg}[i]$ contains the degree of node i.

In order to conserve machine memory, we store these data structures in a disk-based key-value store—our implementation uses Google leveldb (`www.leveldb.org`), but any other disk database should work as well. Although storing the data on hard disks incurs significant read latencies in principle (around 10 milliseconds per read), we have observed that leveldb's memory cache amortizes latencies across multiple reads very well, and therefore disk-based access is not a bottleneck for our implementation of the SVI algorithm. In our large-scale experiments, leveldb could sustain 100s of thousands of reads per second per machine

(using multiple worker threads) on a standard hard drive. This throughput is more than sufficient for the SVI algorithm for STM. The use of disk-based storage frees up roughly 40GB of memory on each machine, when we run our implementation of the SVI algorithm on the 101-million-node web graph. Note that converting an edge list of a network with M edges to these disk-based data structures only costs $\mathrm{O}(M)$ (for hash-based dictionaries) or $\mathrm{O}(M \log M)$ (for tree-based dictionaries) work, and in practice this step takes only a small fraction of the time required by the SVI algorithm.

Subsampling CN Triangles from the Implicit Representation. We employ the following procedure to subsample a mini-batch S of CN triangles (step (1) in Algorithm 1), based on the aforementioned disk-based data structures.

`STM-SUBSAMPLE`**:**

- **For each node** $i = 1$ **to** N **in parallel:**
  - **For** $c = 1$ **to** C**:**
    1. Let $D_i := \texttt{Deg}[i]$.
    2. Draw two *distinct* indices $a, b \in \{1, \ldots, D_i\}$ uniformly at random.
    3. Let $j := \texttt{Adj}[(i, a)]$ and $k := \texttt{Adj}[(i, b)]$.
    4. Check if $(i, j) \in \texttt{Edg}$. If yes, output (i, j, k) as a 3-edge triangle. If no, output (i, j, k) as a 2-edge triangle centered at node i.

Because there is no need to keep all local variational parameters after accumulating all natural gradient sufficient statistics for the global parameters $\boldsymbol{\eta}, \boldsymbol{\gamma}$, we discard the mini-batch S after step (3) of Algorithm 1 in order to save memory. By combining implicit triangle representation and subsampling strategy in this manner, our SVI algorithm for STM avoids having to store trillions of triangles in an explicit form, thus saving TBs (or more) of disk storage and memory space.

Discussion of Subsampling Strategy. Our fast parallel procedure to subsample a mini-batch S of CN triangles may not provide an unbiased estimate of the natural gradient of the global parameters $\tilde{\nabla}\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$ (Equation 6). The reason is that the CN triangles in S are not necessarily sampled from the space of all triangles uniformly at random—one can see that triangles adjacent to low-degree nodes tend to be more likely to be sampled than triangles attached to high-degree nodes. However, it is worth emphasizing that since low-degree neighbors of a high-degree node are very likely to sample triangles involving that high-degree node, the high-degree nodes are still well represented in the set of subsampled triangles. While this bias due to nonuniform sampling can be corrected by appropriately reweighting the subsampled triangles[8], we have counterintuitively observed that community detection performance is actually *more accurate* without the correction factor. Our hypothesis is that, by not reweighting the triangles, low-degree nodes are given more attention because our subsampling procedure is more likely to select their adjacent triangles, compared to uniform triangle sampling. This improves community detection accuracy on low-degree

8. One reweighting scheme is to divide each triangle's natural gradient contribution by its actual probability of being sampled. Similar techniques were employed to reweight the subsampled node pairs in the assortative MMSB inference algorithm (Gopalan et al., 2012; Gopalan and Blei, 2013).

nodes, which in turn improves overall accuracy because the majority of nodes in real-world scale-free networks are low-degree. For this reason, we choose to not reweight the triangles in our experiments.

Earlier development on triangular modeling (Ho et al., 2012c; Yin et al., 2013) advocated a triangle pre-selection technique called δ-subsampling, where a proper subset of all triangles is subsampled *prior to* inference. Our distributed-parallel SVI algorithm for STM differs in that we *never* pre-select triangles, allowing the inference algorithm to (in principle) access all triangles—thus eliminating one source of approximation. It also avoids the need to tune the parameter δ, the number of triangles to pre-select. From a practical standpoint, our new subsampling strategy is far more memory-efficient, as triangles are immediately discarded after use.

5.2 Handling Big STM Models with Over 100 Billion Parameters

At first estimation, the variational parameters γ_i's for all the mixed-membership vectors θ_i's require NK floating point values to be stored. If $N = 100$ million and $K = 1000$, we would need 100 billion 4-byte floating point values (400GB)[9]. Although the Petuum parameter server discussed in the next subsection can evenly distribute the memory load over all participating machines, it requires additional memory to cache for high performance because the parameter server is a caching system. Thus, the actual memory requirement of SVI algorithm for STM is much more than $NK \times 4$ bytes—we have observed that 400GB of γ_i's would require multiple TBs of memory across all machines.

To reduce memory usage to practical levels, we exploit the observation that most nodes in real networks belong to just a few communities, as seen in ground-truth data (Yang and Leskovec, 2012b). In other words, the mixed-membership vectors θ_i's and the corresponding variational parameters γ_i's are extremely sparse. Therefore, we use dictionary data structures to store γ_i's, and perform two approximations at every iteration of the SVI algorithm in order to reduce the memory requirement:

1. We delete elements of γ_i's that are already close to zero—defined as being less than $2\alpha = 0.2$ in the ground-truth experiments (Section 6). See Section 7.1 for an alternative threshold value used for a 101-million-node web graph.

2. During the global update for each γ_i (Equation 7), we only commit the U_0 largest vector elements in the natural gradient of γ_i and set the rest of elements to be zeros. We use $U_0 = 10$ in our experiments, because empirically most nodes exhibit no more than 10 communities and thus it suffices to retain the 10 most significant communities per node.

When the SVI algorithm terminates, we also output γ_i's in a sparse format in order to conserve hard disk space. Experimentally, we have observed that these approximations incur little impact on community detection accuracy, and they allow us to analyze a 101-million-node network with 1000 communities, using only 500GB of memory spread across 5 machines (i.e., 100GB per machine, which is readily available from cloud compute providers such as Amazon EC2).

9. Note that there are only $O(K)$ variational parameters $\boldsymbol{\eta}$ for the triangle-generating probability tensor $\mathbf{B}$, which is extremely small compared to the size of γ_i's.

5.3 Overcoming Slow Network Interfaces

Our distributed-parallel SVI algorithm for STM (Algorithm 1) adopts a data-parallel scheme: each worker thread in the cluster is responsible for a disjoint set of nodes and all N nodes are collectively covered by all workers, in order to update a single set of shared global parameters γ and η. This requires workers to frequently synchronize all parameter updates to each other. However, due to the higher latency and lower bandwidth of computer network interfaces (e.g., 1Gb or 10Gb Ethernet) compared to internal CPU-to-RAM communication, the workers are often forced to wait for communication to complete, thus wasting as much as $\geq 80\%$ of CPU power (Ho et al., 2013).

In order to alleviate this communication bottleneck, various bounded-asynchronous parameter synchronization systems have been developed (Ho et al., 2013; Li et al., 2013; Dai et al., 2015), which essentially reduce the frequency of communication so as to allow for much higher CPU utilization. The trade-off is that reduced communication leads to *staleness* (i.e., out-of-date values) in the workers' view of the model parameters, which could incur errors in algorithm execution. By using the right "consistency models", most iterative optimization and sampling-based algorithms can be made to theoretically and empirically converge despite staleness. Furthermore, while having stale values of the model parameters could decrease convergence progress per iteration, the increase in CPU utilization more than makes up by enabling more iterations per minute, thus yielding much faster distributed machine learning algorithm execution.

These considerations apply to our SVI algorithm for STM as well, because the global variational parameters γ and η are subject to frequent additive updates from all workers (Equation 7) due to the relatively small mini-batch sizes being used (CN triangles per iteration with $C = 1$ in our experiments). To overcome these challenges, we develop our C++ implementation of the SVI algorithm on top of the Petuum system for scalable distributed machine learning (Ho et al., 2013; Lee et al., 2014; Dai et al., 2015), using its bounded-asynchronous parameter server for data-parallel machine learning programming. Petuum features an machine-learning-specific "consistency model", Stale Synchronous Parallel (SSP), that exploits the concept of *bounded staleness* to speed up distributed computation while still providing theoretical guarantees on the convergence of various machine learning algorithms (e.g., stochastic gradient descent). More formally, a worker reading parameters at iteration c will see the effects of all updates from iteration 0 to $c - s - 1$, where $s \geq 0$ is a user-controlled staleness threshold. In our experiments, we found that configuring the SSP staleness setting to either $s = 0$ or $s = 1$ yielded very good, near-linear scaling from 1 through 5 machines (note that 5 machines are sufficient for performing community detection on a 101-million-node network).

6. Empirical Study on Networks with Ground-truth Communities

In this section, we evaluate the overlapping community detection performance of the SVI algorithm for STM, comparing it to both probabilistic and non-probabilistic baseline algorithms. It is also possible to use our algorithm to perform link prediction, although we do not pursue it in this paper. See Yin et al. (2013) for an appropriate procedure.

Network	# nodes N	# edges M	# communities	Edge density M/N^2	Fraction of closed triangles
DBLP	317,080	1,049,866	13,477	1.044×10^{-5}	0.1283
Amazon	334,863	925,872	75,149	8.257×10^{-6}	0.07925
Youtube	1,134,890	2,987,624	8,385	2.320×10^{-6}	0.002081
Livejournal	3,997,962	34,681,189	287,512	2.170×10^{-6}	0.04559

Table 3: Basic statistics of the networks with ground-truth overlapping communities used in our evaluation. These networks are available at `http://snap.stanford.edu/data/`.

6.1 Ground-truth Data

Because there has been debate over the appropriateness of simulated networks in the community detection task (Leskovec et al., 2009; Yang and Leskovec, 2012b), we performed all our experiments on real-world networks with ground-truth communities provided by Yang and Leskovec (2012b), rather than using synthetic benchmarks such as Lancichinetti-Fortunato-Radicchi (LFR) (Lancichinetti and Fortunato, 2009). The ground-truth communities were constructed based on the publicly available meta-data associated with each network (e.g., self-declared interests, hobbies, and affiliations in social networks such as LiveJournal). The size of these networks ranges from $N \approx 300000$ to 65 million, and the number of created communities ranges from $K \approx 8000$ in a 1.1-million-node network to $K \approx 6.3$ million in a 3-million-node network. However, not all of the communities were well-defined. Thus, Yang and Leskovec (2012b) also provided the top 5000 communities with highest quality in each network, where the quality of a community is measured by averaging over several well-known community scoring functions such as conductance, modularity, and triangle-participation-ratio—the idea being that communities that score well on all functions are very likely to be of high quality. We do not claim these ground-truth communities are the only valid way to decompose the network, and acknowledge that there may exist alternative ways to extract plausible communities from large networks.

Table 3 provides basic statistics that are taken from the original directed form of each network in this evaluation. When our algorithm loaded a network, it converted the network to its undirected form via symmetrization because our STM is a probabilistic network model for undirected triangular motifs. When running those baselines that are able to to exploit directed form, we always provided the original directed network as input.

6.2 Evaluation by Normalized Mutual Information (NMI)

We used the normalized mutual information (NMI) (Lancichinetti and Fortunato, 2009), one of the most widely used measures, for evaluating the quality of discovered overlapping communities. The NMI is on a scale of 0 to 1, with 1 corresponding to a perfect matching with the ground-truth communities. In these ground-truth networks, the top 5000 communities were provided by Yang and Leskovec (2012b) as *hard assignments* of nodes: for a particular community, each node is either in that community or it is not (i.e., no partial membership). Two issues must be addressed before the outputs of different algorithms can be evaluated against the ground truth.

First, because the SVI algorithm for STM outputs the variational parameters γ_i's that are then normalized to produce continuous-valued mixed-membership vectors to represent soft community assignments, we must threshold to obtain hard community assignments

that can be compared to the ground truth. Based on the observation that more than 90% of nodes have no more than 10 ground-truth communities, we chose the threshold value to be 0.1, i.e., community k contains node i if $\hat{\theta}_{i,k} = \gamma_{i,k} / \sum_{k'} \gamma_{i,k'} \geq 0.1$. This allows us to detect up to 10 communities per node. We also applied this thresholding procedure to any baselines that output soft community assignments; baselines that output hard assignments were left as it is.

The second issue is that the top 5000 ground-truth communities only cover a small fraction of the network—in other words, the majority of nodes have missing ground-truth community assignments. However, all the algorithms being evaluated assign every node to at least one community, and the NMI cannot handle missing community assignments. Thus, for every node i without a ground-truth assignment in the top 5000 communities, we removed node i from its corresponding communities discovered by each algorithm. This ensures that the NMI is computed only on nodes found within the top 5000 ground-truth communities. We emphasize that such post-processing is fair because it is performed for all the algorithms being evaluated.

6.3 Experimental Settings

Termination Criterion. We monitored the convergence of the SVI algorithm by computing the variational mini-batch lower bound $\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$ (Equation 6) at each iteration, which serves as an unbiased approximation to the true variational lower bound $\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$ (Equation 5). We never compute the true lower bound as it involves all triangles in the network, which would be computationally prohibitive. In our ground-truth experiments, we terminated the algorithm when $\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$ decreases for the first time. This signifies that the SVI algorithm is beginning to oscillate, and will not make much further progress. All our trials on ground-truth networks terminated with high-quality results within 200 iterations under this criterion (using $C = 1$ triangle subsampled for each node, per iteration); in other words, the SVI algorithm only had to process $< 200N$ triangles in total. The intuition is that node community memberships are empirically sparse—each node is unlikely to participate in more than a few communities, and thus a small number of subsampled adjacent triangles suffices to determine its community membership.

Baselines. For comparison with the SVI algorithm for STM, we selected baselines that (1) are able to perform overlapping community detection and (2) are scalable enough to analyze 1-million-node networks in at most a few days. These two criteria greatly narrow the list of candidate baselines, and eventually we considered the following four algorithms:

1. `a-MMSB`: the assortative MMSB (Gopalan et al., 2012; Gopalan and Blei, 2013), which requires the number of communities K to be input as a parameter. We used the single-machine C code available at `https://github.com/premgopalan/svinet`, and applied the "link-sampling" subsampling scheme, as recommended by the user manual.

2. `PM`: the Poisson model (Ball et al., 2011), which requires the number of communities K to be input as a parameter. We used the single-machine C code provided by the authors.

3. `SLPA`: the speaker-listener label propagation algorithm (SLPA) (Xie and Szymanski, 2012), which can automatically detect the number of communities in the network,

given a threshold $r \in [0, 1]$. We found that the NMI score did not vary significantly with different choices of r, so we fixed $r = 0.25$ in all experiments. The single-machine Java code is available at `https://sites.google.com/site/communitydetectionslpa/ganxis` under the name GANXiS.

4. `SVD+M`: a baseline that first applies a rank-K SVD to the adjacency matrix and then extracts communities from the singular vectors using modularity as a stopping criterion, as proposed in Prakash et al. (2010) and Kang et al. (2011). As code was unavailable from the authors, we wrote our own single-machine MATLAB implementation, based on the multi-threaded `svds()` sparse low-rank SVD function.

Both a-MMSB and PM are mixed-membership models based on adjacency matrix representation of networks; SLPA is a message-passing algorithm, in which community labels are propagated along edges until convergence; SVD+M is a matrix factorization algorithm. We ran all algorithms with their default settings, unless otherwise stated. For algorithms that require the number of communities K as input (including our SVI algorithm for STM), we repeated the experiments for different values of K: 5000, 10000, 15000 and 20000.

Machine Configuration. We used server machines equipped with 128GB RAM and 2 Intel Xeon E5-2450 8-core processors, for a total of 16 CPU cores per machine running at 2.10GHz. We ran the distributed-parallel SVI algorithm using 4 such machines, for a total of 64 cores/worker threads and 512GB distributed RAM. The baselines a-MMSB, PM, and SLPA are all single-machine and single-threaded, while SVD+M is single-machine but multithreaded; we ran all algorithms using one machine with 128GB RAM.

6.4 Experimental Results

We first investigate how the mini-batch size CN influences the performance of the SVI algorithm for STM. Then we compare it to various baseline algorithms in terms of NMI score, runtime, and size of detected communities.

Why Choose $C = 1$? Table 4 shows NMI scores and time to convergence for running the SVI algorithm with $C = 1$ v.s. $C = 10$ on three networks. Note that using $C = 1$ is 5-10 times faster than $C = 10$, while maintaining comparable NMI scores. Accordingly, we set the mini-batch size to $C = 1$ triangle per node, resulting in a total of N triangles per iteration.

NMI Score and Runtime. Table 5 shows NMI scores and runtimes for all the algorithms being evaluated. Performing overlapping community detection at these large scales is extremely computationally intensive, and our distributed-parallel SVI algorithm for STM finished execution in far less time than the baselines. None of the baseline algorithms were able to finish the 4-million-node, 35-million-edge Livejournal network within our experimental limit of 5 days, and the SVD+M algorithm was not able to finish any experiment within this limit[10]. Given a larger compute cluster, we expect our approach to finish more rapidly,

10. The main reason is that high-rank SVD is expensive to compute in MATLAB. The call to `svds()` alone took 4 days to complete on the smallest DBLP network ($N = 317$K) for just $K = 5000$, despite being a multithreaded implementation. Furthermore, `svds()` simply ran out of memory on larger experiments, despite equipped with 128GB RAM.

	NMI		Time to convergence	
Network	$C = 1$	$C = 10$	$C = 1$	$C = 10$
DBLP ($N = 317$K)				
$K = 5000$	0.439	0.451	15min	2.8h
$K = 10000$	0.506	0.505	18min	1.5h
$K = 15000$	0.542	0.535	26min	2.1h
$K = 20000$	0.559	0.553	30min	3.1h
Amazon ($N = 334$K)				
$K = 5000$	0.790	0.791	38min	3.4h
$K = 10000$	0.758	0.771	18min	1.4h
$K = 15000$	0.743	0.752	24min	2.2h
$K = 20000$	0.733	0.739	31min	3.2h
Youtube ($N = 1.1$M)				
$K = 20000$	0.433	0.434	7.6h	93h

Table 4: NMI scores and time to convergence for running the SVI algorithm for STM with $C = 1$ v.s. $C = 10$ on three networks. Using $C = 1$ is nearly as accurate as $C = 10$, while requiring much less computational time.

allowing even larger networks to be analyzed in a matter of hours. The STM SVI algorithm is also memory-efficient, due to sparse data structures in the Petuum parameter server used for sharing the global variational parameters γ_i's. In particular, the largest experiment on the Livejournal network with $K = 20000$ only required 24GB RAM on each of the 4 machines (for a total of 96GB). The a-MMSB inference algorithm implementation does not use sparse storage, which caused it to run out of memory on relatively small experiments, even on a machine with 128GB RAM. We believe this underscores the importance of sparse memory management for large-scale problems (Section 5.2).

In terms of accuracy of recovering ground-truth overlapping communities, the STM SVI algorithm outperformed other mixed-membership models a-MMSB and PM, but had lower NMI scores than SLPA (a message-passing algorithm that is not based on a statistical model). This suggests that mixed-membership network models have some room for improvement. However, there are still two advantages of the STM SVI algorithm compared to SLPA: (1) the STM SVI algorithm finishes execution in far less time on a distributed compute cluster, allowing it to scale to much larger networks; (2) SLPA only outputs hard, binary community assignments, so it may not be suitable for analyses that require soft, probabilistic assignments, which the STM SVI algorithm can provide.

Though faster execution of our STM SVI algorithm hinges on the distributed implementation, we would like to point out that even on smaller networks (N ranges from 10000 to 200000) in the absence of distributed computing, our earlier development (Ho et al., 2012c; Yin et al., 2013) has demonstrated the benefits of triangular modeling over adjacency-matrix-based modeling in terms of scalability and accuracy. The single-machine SVI algorithm on these smaller networks usually converges after several passes on all triangles in the network (4-5 passes at most), and achieves competitive or improved accuracy for latent space recovery and link prediction compared to MMSB (Yin et al., 2013). This is compatible with our observation that, in the distributed setting, at most $200N$ triangles needed to be subsampled and processed to achieve the indicated NMI scores in Table 5.

	NMI					Time to completion				
Network	STM SVI	a-MMSB	PM	SLPA	SVD+M	STM SVI	a-MMSB	PM	SLPA	SVD+M
DBLP ($N = 317$K)										
$K = 5000$	0.439	0.379	0.251	0.581	DNF	15min	17.4h	8.9h	2.6h	> 5 days
$K = 10000$	0.506	0.437	0.294	(16874 coms)	DNF	18min	48h	18h		> 5 days
$K = 15000$	0.542	OOM	0.322		OOM	26min	OOM	51h		OOM
$K = 20000$	0.559	OOM	0.341		OOM	30min	OOM	96h		OOM
Amazon ($N = 335$K)										
$K = 5000$	0.790	0.750	0.483	0.867	DNF	38min	31h	4.4h	3.2h	> 5 days
$K = 10000$	0.758	OOM	0.548	(29021 coms)	DNF	18min	OOM	11h		> 5 days
$K = 15000$	0.743	OOM	0.571		OOM	24min	OOM	22h		OOM
$K = 20000$	0.733	OOM	0.576		OOM	31min	OOM	31h		OOM
Youtube ($N = 1.1$M)										
$K = 5000$	0.374	OOM	0.129	0.424	OOM	2.2h	OOM	76h	21h	OOM
$K = 10000$	0.422	OOM	DNF	(5972 coms)	OOM	3.2h	OOM	> 5 days		OOM
$K = 15000$	0.456	OOM	DNF		OOM	3.9h	OOM	> 5 days		OOM
$K = 20000$	0.433	OOM	DNF		OOM	7.6h	OOM	> 5 days		OOM
Livejournal ($N = 4.0$M)										
$K = 20000$	0.743	OOM	DNF	DNF	OOM	63h	OOM	> 5 days	> 5 days	OOM

Table 5: NMI scores and runtimes for all the algorithms being evaluated. "OOM" means the algorithm ran out of memory and failed, while "DNF" means the algorithm did not finish within a reasonable amount of time (5 days of continuous computation). Note that the SLPA algorithm can automatically select the number of communities K, thus we report only one NMI score and runtime per network (with the number of detected communities in parentheses).

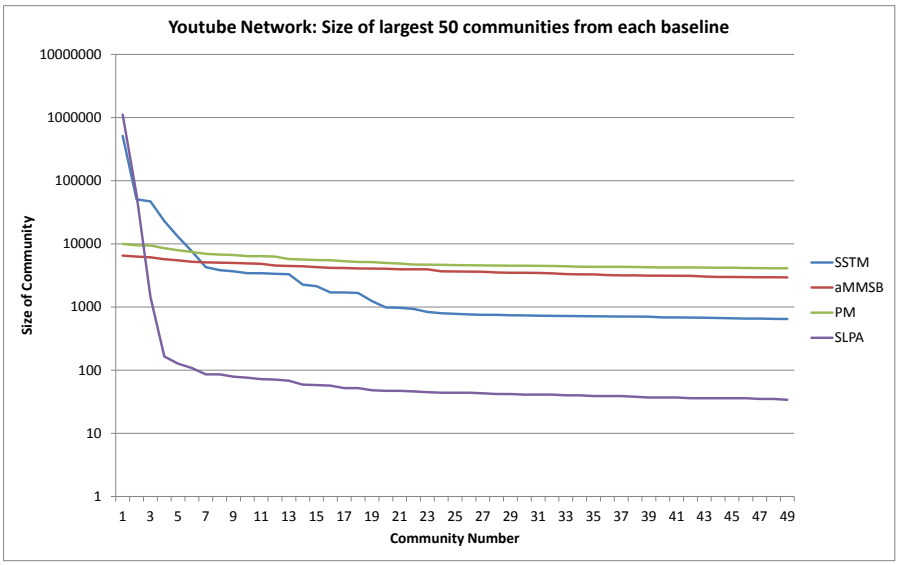


Figure 5: Youtube network: size of the largest 50 detected communities by STM SVI, a-MMSB, PM, and SLPA, plotted on a logarithmic scale.

Size of Detected Communities. Figure 5 plots the size of the largest 50 detected communities in the Youtube network by STM SVI, a-MMSB, PM, and SLPA, on a logarithmic scale. For algorithms that require the number of communities K as input, we set $K = 1000$. It was necessary for us to use the Youtube network with $K = 1000$ communities, in order to obtain a plot of community size for a-MMSB; otherwise, it ran out of memory (see Table 5). We observed that STM SVI and SLPA were able to detect several very large communities containing over 50K nodes, whereas a-MMSB and PM could not recover communities with more than 10K nodes. Furthermore, the vast majority of communities detected by STM SVI and SLPA are much smaller than the ones discovered by a-MMSB and PM. In other words, the distributions of STM SVI and SLPA community size are far more skewed (power-law-like) than a-MMSB's and PM's. We hypothesize that the better NMI performance of STM SVI and SLPA is partly due to their ability to detect very large and very small communities, which are present in real-world networks with power-law behavior (Leskovec et al., 2009). We also conjecture that because larger communities in real-world networks are likely to be edge-sparse, with intra-community edge probability p not much larger than the inter-community edge probability q (Leskovec et al., 2009), it would be theoretically difficult for models such as a-MMSB to detect these large communities (Anandkumar et al., 2014).

7. Empirical Study on a 101-million-node Web Graph

Our SVI algorithm for STM is intended for latent space inference in very large, Internet-scale networks. In that vein, we conclude with a brief analysis of the 101-million-node, 2-billion-edge Subdomain/Host web graph, from the Web Data Commons (available at `http://webdatacommons.org/hyperlinkgraph/`). This web graph was constructed from the Common Crawl 2012 web corpus, which originally contained 3.5 billion web pages and 128

billion hyperlinks. The 3.5 billion web pages were aggregated by their subdomain[11] or host, resulting in a new graph with 101 million nodes/subdomains. An edge was created between two subdomains if at least one hyperlink was found between their aggregated pages, resulting in 2 billion directed edges. When we ran our algorithm, it was treated as an undirected, unweighted network via symmetrization.

Our primary aim is to demonstrate that the our distributed-parallel SVI algorithm for STM finished execution on this massive web graph in an acceptable amount of time on a small cluster—37.3 hours using $K = 1000$ overlapping communities. This scale is 25 times larger than a recent experiment by Gopalan and Blei (2013), who ran the a-MMSB inference algorithm on an $N \approx 4$M patent network using $K = 1000$. We also performed light qualitative analysis to show that the discovered mixed-membership vectors $\hat{\theta}_i$'s reveal sensible insights about the structure of the web graph.

7.1 Experimental Settings

We used the experimental settings that were mostly similar to the experiments on ground-truth networks, with the following exceptions:

Number of Machines. We used 5 machines with 16 CPU cores and 128GB RAM per machine, configured identically to the 4 machines used in the ground-truth experiments. The extra machine was required because the algorithm ran out of memory on 4 machines.

Termination Criterion. On such a large network, the SVI algorithm may take a long time before the variational mini-batch lower bound starts oscillating, which is the termination criterion used in the ground-truth experiments. For this web graph, we observed that by iteration $t = 50$, the lower bound was only changing at the 5th significant figure (relative to the first few iterations). We thus concluded that the algorithm was no longer making significant progress, and stopped the algorithm at $t = 50$ iterations.

Threshold of γ_i's to Maintain Sparsity. As explained in Section 5.2, this massive web graph requires 100 billion floating point values for the variational parameters γ_i's, which is estimated at TBs of memory if stored densely (after accounting for algorithmic overheads). In order to keep the γ_i's sparse enough to fit on 5 machines with 640GB total RAM, we zeroed out any element $\gamma_{i,k} < 10$ at the end of every iteration. Though this is more aggressive than the threshold $2\alpha = 0.2$ used in the ground-truth experiments, it has little impact on the detected communities because of the way we obtain the final hard community assignments based on the inferred $\hat{\theta}_i$'s (Section 6.2). When the SVI algorithm terminated, the vast majority of nodes i had an unnormalized "total variational mass" $\sum_k \gamma_{i,k} > 100$. Thus, any zeroed-out element $\gamma_{i,k}$ would have been normalized to $\hat{\theta}_{i,k} = \gamma_{i,k}/(\sum_{k'} \gamma_{i,k'}) < 0.1$, which is below the threshold of being detected as part of community k.

7.2 Qualitative Analysis

To decide the size of detected communities, we treated the mixed-membership vectors $\hat{\theta}_{i,k}$ as an $N \times K$ continuous-valued matrix. The column sum of the kth column accounts for

11. A subdomain is an Internet host name with 3 or more levels. For example, `www.cmu.edu` and `www.ml.cmu.edu` are considered to be two distinct subdomains.

all partial memberships in community k, and thus is regarded as the effective number of nodes in that community.

To identify significant nodes associated with each community k, we computed a score $s_k(i) = \hat{\theta}_{i,k} \times D_i$ for each node i—its partial membership in community k multiplied by its node degree—and then sorted $s_k(1), \ldots, s_k(N)$ in descending order. Table 6 shows the top-scoring 10 nodes from each of the largest 5 communities, as well as the estimated fraction of 3-edge triangles in each community (i.e., the parameter $B_{kkk,2}$ in Table 2). The largest community is a giant core dominated by well-known websites including youtube.com, google.com, and twitter.com, with a much lower fraction of 3-edge triangles (0.062) than the other communities. The 2nd to 5th largest communities are:

- Community 2: mostly "online stores" that focus on specific products (kitchenware, phones, or cars). However, these are not real online stores as the entries all link to eBay auction pages. The community memberships $\hat{\theta}_{i,k}$ of the top 10 nodes are all perfect (1.0), which suggests that these sites (1) form a compact community with few outside links and (2) are probably copies of each other. It is likely that these sites are meant to increase the visibility of certain eBay auctions—essentially, a form of search engine optimization (SEO). It is unclear whether the auctions themselves are fraudulent or not.

- Community 3: dominated by webpages from Lycos/Tripod, two related companies in the search and website creation business. It also contains w3schools.com, a site that provides instructions for website creation. The top websites in this community do not appear suspicious.

- Community 4: Spanish-language websites, particularly Hispavista, an Internet firm based in Spain. The top websites in this community do not appear suspicious.

- Community 5: dominated by blackmagic.org/com, a website that maintains a large list of websites crawled from the Internet in 2006. It is likely that blackmagic.org/com acts as a hub for the other nodes in the community, which all have much smaller scores $s_k(i)$. The extremely high fraction of 3-edge triangles (0.985) suggests that this community is close to a full clique, or perhaps several cliques connected by one or two bridge nodes. Previous works have shown that such clique-like patterns in web graphs are highly indicative of web duplication or fraud (Kang et al., 2009, 2011); a cursory look at the websites in this community (such as the skimium.* nodes) reveals that many of them are near-exact copies of each other.

We also explored significant nodes that are in 2 or more communities in this web graph (recall that a node i is assigned to community k if $\hat{\theta}_{i,k} \geq 0.1$). Of these 700K websites, the one with the highest node degree, wordpress.org, has a partial membership 0.89 in the giant core and 0.11 in the 25th largest community. By examining the top websites in the 25th largest community, we found several domains of hostgator.com, a website hosting business. Further inspection revealed that the wordpress forums frequently recommend hostgator.com to host wordpress-powered blogs, and similarly, the hostgator support pages explain how to set up a wordpress blog on hostgator itself.

k-th largest community	Fraction of 3-edge triangles	Website	$\hat{\theta}_{i,k}$	$s_k(i)$
1	0.062	youtube.com	0.96	2906030.5
(mass 71038.3K)		wordpress.org	0.89	2102190.9
		en.wikipedia.org	0.93	1899359.3
		gmpg.org	0.92	1638740.7
		tumblr.com	0.94	1096803.3
		twitter.com	0.95	1057107.2
		flickr.com	1.00	931931.7
		serebella.com	1.00	757930.4
		google.com	0.92	738368.1
		top20directory.com	1.00	691007.1
2	0.604	kitchensnstuff.com	1.00	163675.0
(mass 165.7K)		shopping.mia.net	1.00	105814.0
		thenichestorebuilder.com	1.00	57502.3
		generator.mia.net	1.00	56772.0
		phone.mia.net	1.00	53360.0
		seekwonder.com	1.00	44767.9
		hostinglizard.com	1.00	44496.5
		corvette-auction.com	1.00	43633.9
		gotomeeting.mia.net	1.00	43611.0
		cashadvance.mia.net	1.00	43587.9
3	0.481	tripod.lycos.com	0.69	697557.6
(mass 88.7K)		w3schools.com	0.99	499886.4
		domains.lycos.com	0.99	476899.0
		club.tripod.com	0.13	63429.6
		wired.com	0.37	46150.2
		search.lycos.com	0.94	29350.7
		news.lycos.com	0.94	20329.7
		moreover.com	0.15	613.1
		metallica.com	0.09	421.4
		google-pagerank.net	0.56	407.8
4	0.695	hispavista.com	0.80	156246.8
(mass 69.7K)		dominios.hispavista.com	0.94	138596.5
		inmobiliaria.hispavista.com	0.93	138045.6
		globedia.com	0.87	134000.4
		galeon.com	0.84	133910.6
		neopolis.com	0.94	133033.7
		trabajos.com	0.89	132235.4
		horoscopo.hispavista.com	0.95	131548.8
		paginasamarillas.hispavista.com	0.95	131476.2
		software.hispavista.com	0.94	131310.0
5	0.985	blackmagic.org	0.60	103660.8
(mass 65.2K)		blackmagic.com	0.60	102844.9
		opera.com	0.04	2265.0
		skimium.fr	0.88	1289.3
		skimium.co.uk	0.97	1181.0
		skimium.es	0.97	1167.6
		skimium.nl	0.96	1140.9
		skimium.it	0.94	1137.8
		skimium.be	0.94	1111.3
		inetgiant.com	0.24	665.6

Table 6: Subdomain/Host web graph: the 10 top-scoring nodes in each of the largest 5 communities by mass (the effective number of nodes in the community). The score $s_k(i)$ is computed as $\hat{\theta}_{i,k} \times D_i$, i.e., the partial membership of node i in community k multiplied by its node degree. For each community, we also report the estimated fraction of 3-edge triangles. A quick look at the top websites in communities 2 and 5 reveals suspicious behavior (see main text for details).

8. Conclusion and Discussion

Massive Internet-scale networks are technically challenging to explore, manipulate, and visualize. One approach to understanding the structural and functional properties of massive networks is to perform latent space inference for overlapping community detection. At the time of writing, there are few inference algorithms that can scale to hundreds of millions of nodes in order to detect thousands of distinct communities, and almost none of them are based on a probabilistic model.

In this work, starting with the mixed-membership class of statistical models as a foundation, we systematically tackle the statistical and computational challenges associated with Internet-scale network inference. Our major contributions include: (1) characterization of the network as a more compact—and arguably more salient for community detection—triangular representation; (2) design of a parsimonious generative model with only $O(K)$ instead of K^2 or K^3 parameters; (3) construction of an efficient structured stochastic variational inference algorithm; (4) a careful consideration of the distributed implementation in order to handle big network data, big network model, and slow inter-machine communication. The resulting distributed-parallel STM SVI algorithm is able to detect 1000 communities from a 100-million-node network in 1.5 days on just 5 cluster machines, and we believe that networks with $N > 1$ billion nodes can be analyzed with a sufficiently large cluster, thus opening the door to Facebook-scale social networks and beyond.

We would like to end with some directions and open issues for future research. One limitation of STM is that it only applies to undirected and unweighted networks (and triangular motifs), although we have demonstrated good community detection NMI scores relative to baselines. We believe that accounting for edge direction and weights could further improve accuracy, which are more common in non-social-network domains such as biology and finance. Sensitivity to initialization is another issue that we would like to address: while the `STM-CANONIZE` procedure worked well in our experiments, we believe that there are more systematic ways to address the issue, such as augmenting the variational inference algorithm with split-merge-like search moves that can change many node community assignments at once. Finally, we believe that triangular modeling could also be extended to other probabilistic network model for overlapping community detection, such as the Community-Affiliation Graph Model (AGM) by Yang and Leskovec (2012a).

Acknowledgments

The authors would like to thank two anonymous reviewers for their helpful comments that improved the manuscript. This work was supported by AFOSR FA9550010247, NIH 1R01GM093156, and DARPA FA87501220324 to Eric P. Xing. Qirong Ho is supported by an A-STAR, Singapore fellowship. Junming Yin is partly supported by a Ray and Stephanie Lane Research Fellowship from CMU and a research grant from the Center for Management Innovations in Health Care at the Eller College of Management.

Appendix A. Details of Stochastic Variational Inference

In this section, we provide details of our SVI algorithm, including the exact form the variational lower bound, the exact and approximate local update equations, and the natural gradients with respect to the global parameters.

A.1 Exact Form of the Variational Lower Bound

The variational lower bound (Equation 3) of the log marginal likelihood of the triangular motifs based on the variational distribution (Equation 1) is

$$\log p(\mathbf{E} \mid \alpha, \lambda) \geq \mathbb{E}_q[\log p(\mathbf{E}, \mathbf{s}, \boldsymbol{\theta}, \mathbf{B} \mid \alpha, \lambda)] - \mathbb{E}_q[\log q(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B})] \doteq \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma}) \tag{8}$$

$$\begin{aligned}
&= \mathbb{E}_q[\log p(B_0 \mid \lambda)] - E_q[\log q(B_0 \mid \eta_0)] + \sum_{x=1}^{K} \Big\{ \mathbb{E}_q[\log p(B_{xx} \mid \lambda)] - \mathbb{E}_q[\log q(B_{xx} \mid \eta_{xx})] \Big\} \\
&+ \sum_{x=1}^{K} \Big\{ \mathbb{E}_q[\log p(B_{xxx} \mid \lambda)] - \mathbb{E}_q[\log q(B_{xxx} \mid \eta_{xxx})] \Big\} + \sum_{i=1}^{N} \Big\{ \mathbb{E}_q[\log p(\theta_i \mid \alpha)] - \mathbb{E}_q[\log q(\theta_i \mid \gamma_i)] \Big\} \\
&+ \sum_{(i,j,k) \in I} \Big\{ \mathbb{E}_q[\log p(s_{i,jk} \mid \theta_i) + \log p(s_{j,ik} \mid \theta_j) + \log p(s_{k,ij} \mid \theta_k)] \Big\} \\
&+ \sum_{(i,j,k) \in I} \Big\{ \mathbb{E}_q[\log p(E_{ijk} \mid s_{i,jk}, s_{j,ik}, s_{k,ij}, \mathbf{B})] - \mathbb{E}_q[\log q(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \phi_{ijk})] \Big\}.
\end{aligned}$$

The first two lines of Equation 8 represent the global terms $g(\boldsymbol{\gamma}, \boldsymbol{\eta})$ that depend only the global variational parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\eta}$, whereas the last two lines are a summation of the local terms $\ell(\phi_{ijk}, \boldsymbol{\gamma}, \boldsymbol{\eta})$, one for each triangle.

A.2 Local Update (Exact and Approximate)

For each sampled triangle (i, j, k) in a mini-batch, the exact local update algorithm updates all the K^3 entries of ϕ_{ijk}, and then renormalizes them to sum to one. The K^3 entries of ϕ_{ijk} can be segregated into three broad categories: ϕ_{ijk}^{xxx}, ϕ_{ijk}^{xxy}, ϕ_{ijk}^{xyz}, which correspond to the following cases:

1. ϕ_{ijk}^{xxx} corresponds to the case that all nodes choose the same community: $s_{i,jk} = s_{j,ik} = s_{k,ij} = x$;
2. ϕ_{ijk}^{xxy} (or ϕ_{ijk}^{xyx}, or ϕ_{ijk}^{yxx}) corresponds to the case that two nodes choose the same community: $s_{i,jk} = s_{j,ik} = x$ and $s_{k,ij} = y$ (with similar variations for ϕ_{ijk}^{xyx} and ϕ_{ijk}^{yxx});
3. ϕ_{ijk}^{xyz} corresponds to the case that all nodes choose different communities: $s_{i,jk} = x$, $s_{j,ik} = y$, $s_{k,ij} = z$.

The update equations for each category are:

1. For $x \in \{1, \ldots, K\}$,

$$\phi_{ijk}^{xxx} \propto \exp\Big\{ \mathbb{E}_q[\log B_{xxx,2}] \mathbb{I}[E_{ijk} = 4] + \mathbb{E}_q[\log(B_{xxx,1}/3)] \mathbb{I}[E_{ijk} \neq 4] + \mathbb{E}_q[\log \theta_{i,x} + \log \theta_{j,x} + \log \theta_{k,x}] \Big\}. \tag{9}$$

2. For $x, y \in \{1, \ldots, K\}$ and $x \neq y$,

$$\phi_{ijk}^{xxy} \propto \exp\Big\{\mathbb{E}_q[\log B_{xx,3}]\mathbb{I}[E_{ijk} = 4] + \mathbb{E}_q[\log B_{xx,2}]\mathbb{I}[E_{ijk} = 3] + \mathbb{E}_q[\log(B_{xx,1}/2)]\mathbb{I}[E_{ijk} = 1 \text{ or } 2] \\ + \mathbb{E}_q[\log \theta_{i,x} + \log \theta_{j,x} + \log \theta_{k,y}]\Big\}. \tag{10}$$

   The update equations for ϕ_{ijk}^{xyx} and ϕ_{ijk}^{yxx} are similar to ϕ_{ijk}^{xxy} (up to a trivial rearrangement of variables), thus we omit their details.
3. For distinct $x, y, z \in \{1, \ldots, K\}$,

$$\phi_{ijk}^{xyz} \propto \exp\Big\{\mathbb{E}_q[\log B_{0,2}]\mathbb{I}[E_{ijk} = 4] + \mathbb{E}_q[\log(B_{0,1}/3)]\mathbb{I}[E_{ijk} \neq 4] + \mathbb{E}_q[\log \theta_{i,x} + \log \theta_{j,y} + \log \theta_{k,z}]\Big\}. \tag{11}$$

The above are the *exact* update equations for ϕ_{ijk}, and they require $\mathrm{O}(K^3)$ run-time per triangle (i, j, k). The update equations for the $\mathrm{O}(K)$ "mixture-of-deltas" approximation (described in Section 4.2) are almost exactly equivalent to the exact $\mathrm{O}(K^3)$ update procedure, with two minor modifications: (1) we simply zero out entries (a, b, c) that are not in the chosen set $\mathcal{A}$, and (2) we renormalize the chosen entries $(a, b, c) \in \mathcal{A}$ amongst themselves so that they sum to 1. This follows because the "mixture-of-deltas" variational distribution is essentially a categorical or multinomial distribution with some elements constrained to be zero.

A.3 Global update

The global update (Equation 7) in our SVI algorithm requires computing the natural gradient $\tilde{\nabla}\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$. For clarity, we decompose $\tilde{\nabla}\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$ over each part of $\boldsymbol{\eta}, \boldsymbol{\gamma}$. The natural gradient with respect to $\boldsymbol{\eta}$ is:

- For $x \in \{1, \ldots, K\}$,

$$\tilde{\nabla}_{\eta_{xxx,1}}\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma}) = \lambda + \frac{m}{|S|}\left[\sum_{(i,j,k)\in S} q_{ijk}(x, x, x)\mathbb{I}[E_{ijk} \neq 4]\right] - \eta_{xxx,1}, \tag{12}$$

$$\tilde{\nabla}_{\eta_{xxx,2}}\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma}) = \lambda + \frac{m}{|S|}\left[\sum_{(i,j,k)\in S} q_{ijk}(x, x, x)\mathbb{I}[E_{ijk} = 4]\right] - \eta_{xxx,2}. \tag{13}$$

- For $x \in \{1, \ldots, K\}$,

$$\tilde{\nabla}_{\eta_{xxx,1}}\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma}) = \lambda + \frac{m}{|S|}\Bigg[\sum_{(i,j,k)\in S}\sum_{y:y\neq x}\Big(q_{ijk}(x, x, y)\mathbb{I}[E_{ijk} = 1, 2] + q_{ijk}(x, y, x)\mathbb{I}[E_{ijk} = 1, 3] \\ + q_{ijk}(y, x, x)\mathbb{I}[E_{ijk} = 2, 3]\Big)\Bigg] - \eta_{xx,1}, \tag{14}$$

$$\tilde{\nabla}_{\eta_{xxx,2}}\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma}) = \lambda + \frac{m}{|S|}\Bigg[\sum_{(i,j,k)\in S}\sum_{y:y\neq x}\Big(q_{ijk}(x, x, y)\mathbb{I}[E_{ijk} = 3] + q_{ijk}(x, y, x)\mathbb{I}[E_{ijk} = 2] \\ + q_{ijk}(y, x, x)\mathbb{I}[E_{ijk} = 1]\Big)\Bigg] - \eta_{xx,2}, \tag{15}$$

$$\tilde{\nabla}_{\eta_{xxx,3}}\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma}) = \lambda + \frac{m}{|S|}\left[\sum_{(i,j,k)\in S}\sum_{y:y\neq x}\Big(q_{ijk}(x, x, y) + q_{ijk}(x, y, x) + q_{ijk}(y, x, x)\Big)\mathbb{I}[E_{ijk} = 4]\right] - \eta_{xx,3}. \tag{16}$$

- For the sole η_0 parameter:

$$\tilde{\nabla}_{\eta_{0,1}} \mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma}) = \lambda + \frac{m}{|S|} \left[\sum_{(i,j,k) \in S} \sum_{(x,y,z): x \neq y \neq z} q_{ijk}(x,y,z) \mathbb{I}[E_{ijk} \neq 4] \right] - \eta_{0,1}, \tag{17}$$

$$\tilde{\nabla}_{\eta_{0,2}} \mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma}) = \lambda + \frac{m}{|S|} \left[\sum_{(i,j,k) \in S} \sum_{(x,y,z): x \neq y \neq z} q_{ijk}(x,y,z) \mathbb{I}[E_{ijk} = 4] \right] - \eta_{0,2}. \tag{18}$$

The natural gradient $\tilde{\nabla} \mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$ with respect to $\boldsymbol{\gamma}$ is, for each $i = 1, \ldots, N$ and $x = 1, \ldots, K$,

$$\begin{aligned} \tilde{\nabla}_{\gamma_{i,x}} \mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma}) = \alpha + \frac{m}{|S|} \Bigg[& \sum_{(j,k):(i,j,k) \in S} \sum_{y,z} q_{ijk}(x,y,z) + \sum_{(j,k):(j,i,k) \in S} \sum_{y,z} q_{jik}(y,x,z) \\ & + \sum_{(j,k):(j,k,i) \in S} \sum_{y,z} q_{jki}(y,z,x) \Bigg] - \gamma_{i,x}. \end{aligned} \tag{19}$$

References

Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.

E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014, 2008.

L. Akoglu, H. Tong, B. Meeder, and C. Faloutsos. PICS: Parameter-free identification of cohesive subgroups in large attributed graphs. In *SIAM International Conference on Data Mining*, pages 439–450, 2012.

S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

A. Anandkumar, R. Ge, D. Hsu, and S. M. Kakade. A tensor approach to learning mixed membership community models. *Journal of Machine Learning Research*, 15:2239–2312, 2014.

R. Balasubramanyan and W. W. Cohen. Block-LDA: Jointly modeling entity-annotated text and entity-entity links. In *SIAM International Conference on Data Mining*, pages 450–461, 2011.

B. Ball, B. Karrer, and M. E. J. Newman. Efficient and principled method for detecting communities in networks. *Physical Review E*, 84(3):036103, 2011.

A. Beutel, A. Kumar, E. E. Papalexakis, P. P. Talukdar, C. Faloutsos, and E. P. Xing. FlexiFaCT: Scalable flexible factorization of coupled tensors on hadoop. In *SIAM International Conference on Data Mining*, pages 109–117, 2014.

P. Bickel, D. Choi, X. Chang, and H. Zhang. Asymptotic normality of maximum likelihood and its variational approximation for stochastic blockmodels. *Annals of Statistics*, 41(4): 1922–1943, 2013.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

L. Bottou. Stochastic learning. *Advanced Lectures on Machine Learning*, pages 146–168, 2004.

L. Cao and L. Fei-Fei. Spatially coherent latent topic model for concurrent segmentation and classification of objects and scenes. In *International Conference on Computer Vision*, pages 1–8, 2007.

J. Chang and D. M. Blei. Relational topic models for document networks. In *International Conference on Artificial Intelligence and Statistics*, pages 81–88, 2009.

W. Dai, A. Kumar, J. Wei, Q. Ho, G. Gibson, and E. P. Xing. High-performance distributed ML at scale through parameter server consistency models. In *AAAI Conference on Artificial Intelligence*, pages 79–87, 2015.

L. Dietz, S. Bickel, and T. Scheffer. Unsupervised prediction of citation influences. In *International Conference on Machine Learning*, pages 233–240, 2007.

Facebook. Anatomy of facebook, January 2013. URL `www.facebook.com/note.php?note_id=10150388519243859`.

W. Fu, L. Song, and E. P. Xing. Dynamic mixed membership blockmodel for evolving networks. In *International Conference on Machine Learning*, pages 329–336, 2009.

B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *International Conference on Computer Vision*, pages 670–677, 2009.

G. Ghoshal, V. Zlatić, G. Caldarelli, and M. E. J. Newman. Random hypergraphs and their applications. *Physical Review E*, 79:066118, 2009.

P. Gopalan and D. M. Blei. Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences*, 110(36):14534–14539, 2013.

P. Gopalan, D. Mimno, S. Gerrish, M. Freedman, and D. M. Blei. Scalable inference of overlapping communities. In *Neural Information Processing Systems*, pages 2249–2257. 2012.

M. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, 1973.

T. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl 1):5228–5235, 2004.

F. Guo, S. Hanneke, W. Fu, and E. P. Xing. Recovering temporally rewiring networks: A model-based approach. In *International Conference on Machine Learning*, pages 321–328, 2007.

Hadoop. Apache Hadoop. `http://hadoop.apache.org/`, 2012.

M. S. Handcock, A. E. Raftery, and J. M. Tantrum. Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A*, 170(2):301–354, 2007.

Q. Ho, L. Song, and E. P. Xing. Evolving cluster mixed-membership blockmodel for time-varying networks. In *International Conference on Artificial Intelligence and Statistics*, pages 342–350, 2011.

Q. Ho, J. Eisenstein, and E. P. Xing. Document hierarchies from text and links. In *International Conference on World Wide Web*, pages 739–748, 2012a.

Q. Ho, A. Parikh, and E. Xing. A multiscale community blockmodel for network exploration. *Journal of the American Statistical Association*, 107(499), 2012b.

Q. Ho, J. Yin, and E. P. Xing. On triangular versus edge representations — towards scalable modeling of networks. In *Neural Information Processing Systems*, pages 2132–2140, 2012c.

Q. Ho, J. Cipar, H. Cui, J.-K. Kim, S. Lee, P. B. Gibbons, G. Gibson, G. R. Ganger, and E. P. Xing. More effective distributed ML via a stale synchronous parallel parameter server. In *Neural Information Processing Systems*, pages 1223–1231, 2013.

P. Hoff, A. Raftery, and M. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460):1090–1098, 2002.

M. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

D. Hunter, S. Goodreau, and M. Handcock. Goodness of fit of social network models. *Journal of the American Statistical Association*, 103(481):248–258, 2008.

U. Kang, C. E. Tsourakakis, and C. Faloutsos. PEGASUS: A peta-scale graph mining system implementation and observations. In *International Conference on Data Mining*, pages 229–238, 2009.

U. Kang, B. Meeder, and C. Faloutsos. Spectral analysis for billion-scale graphs: Discoveries and implementation. In *Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 13–25, 2011.

C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI Conference on Artificial Intelligence*, pages 381–388, 2006.

D. Krackhardt and M. Handcock. Heider vs Simmel: Emergent features in dynamic structures. In *Conference on Statistical Network Analysis*, pages 14–27, 2007.

A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80 (1):016118, 2009.

C. Lee, F. Reid, A. McDaid, and N. Hurley. Detecting highly-overlapping community structure by greedy clique expansion. In *Workshop on Social Network Mining and Analysis held in Conjunction with the International Conference on Knowledge Discovery and Data Mining*, pages 33–42, 2010.

S. Lee, J. K. Kim, X. Zheng, Q. Ho, G. A Gibson, and E. P. Xing. Primitives for dynamic big model parallelism. *CoRR*, abs/1406.4580, 2014.

J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 177–187, 2005.

J. Leskovec, K. J Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

M. Li, L. Zhou, Z. Yang, A. Li, F. Xia, D. G. Andersen, and A. Smola. Parameter server for distributed machine learning. In *NIPS Workshop on Parallel and Large-scale Machine Learning*, 2013.

Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. GraphLab: A new parallel framework for machine learning. In *Uncertainty in Artificial Intelligence*, 2010.

Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Distributed GraphLab: A framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727, 2012.

K. Miller, T. Griffiths, and M. I. Jordan. Nonparametric latent feature models for link prediction. In *Neural Information Processing Systems*, pages 1276–1284. 2009.

R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

M. Morris, M. Handcock, and D. Hunter. Specification of exponential-family random graph models: terms and computational aspects. *Journal of Statistical Software*, 24(4):1548, 2008.

R. Nallapati, A. Ahmed, E. P. Xing, and W. W. Cohen. Joint latent topic models for text and citations. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 542–550, 2008.

M. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

M. Newman and J. Park. Why social networks are different from other types of networks. *Physical Review E*, 68(3):036122+, 2003.

G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

J. Parkkinen, J. Sinkkonen, A. Gyenge, and S. Kaski. A block model suitable for sparse graphs. In *International Workshop on Mining and Learning with Graphs*, 2009.

B. A. Prakash, A. Sridharan, M. Seshadri, S. Machiraju, and C. Faloutsos. EigenSpokes: Surprising patterns and scalable community chipping in large graphs. In *Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 435–448, 2010.

I. Psorakis, S. Roberts, M. Ebden, and B. Sheldon. Overlapping community detection using Bayesian non-negative matrix factorization. *Physical Review E*, 83(6):066114, 2011.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Uncertainty in Artificial Intelligence*, pages 487–494, 2004.

M. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13 (7):1649–1681, 2001.

G. Simmel and K. Wolff. *The Sociology of Georg Simmel*. Free Press, 1950.

A. Singh and G. Gordon. A bayesian matrix factorization model for relational data. In *Uncertainty in Artificial Intelligence*, pages 556–563, 2010.

D. Stasi, K. Sadeghi, A. Rinaldo, S. Petrovic, and S. Fienberg. β models for random hypergraphs with a given degree sequence. In *International Conference on Computational Statistics*, 2014.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding. Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery*, 22(3):493–521, 2011.

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

J. Xie and B. Szymanski. Towards linear time overlapping community detection in social networks. In *Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 25–36, 2012.

J. Xie, S. Kelley, and B. Szymanski. Overlapping community detection in networks: the state of the art and comparative study. *ACM Computing Surveys*, 45(4), 2013.

Yahoo. Webscope from yahoo! labs, January 2013. URL `http://webscope.sandbox.yahoo.com/catalog.php?datatype=g`.

J. Yang and J. Leskovec. Community-affiliation graph model for overlapping network community detection. In *International Conference on Data Mining*, pages 1170–1175, 2012a.

J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *ACM SIGKDD Workshop on Mining Data Semantics*, pages 3:1–3:8, 2012b.

J. Yin, Q. Ho, and E. P. Xing. A scalable approach to probabilistic latent space inference of large-scale networks. In *Neural Information Processing Systems*, pages 422–430. 2013.

M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: cluster computing with working sets. In *USENIX Conference on Hot Topics in Cloud Computing*, 2010.