2014 IEEE International Conference on Data Mining

A Fast Inference Algorithm for Stochastic Blockmodel

Zhiqiang Xu
Nanyang Technological University
Singapore
zxu1@e.ntu.edu.sg

Yiping Ke
Nanyang Technological University
Singapore
ypke@ntu.edu.sg

Yi Wang
Institute of High Performance Computing
A*STAR, Singapore
wangyi@ihpc.a-star.edu.sg

***Abstract*—Stochastic blockmodel is a widely used statistical tool for modeling graphs and networks. Despite its popularity, the development on efficient inference algorithms for this model is surprisingly inadequate. The existing solutions are either too slow to handle large networks, or suffer from convergence issues. In this paper, we propose a fast and principled inference algorithm for stochastic blockmodel. The algorithm is based on the variational Bayesian framework, and deploys the natural conjugate gradient method to accelerate the optimization of the variational bound. Leveraging upon the power of both conjugate and natural gradients, it converges superlinearly and produces high quality solutions in practice. In particular, we apply our algorithm to the community detection task and compare it with the state-of-the-art variational Bayesian algorithms. We show that it can achieve up to two orders of magnitude speedup without significantly compromising the quality of solutions.**

I. Introduction

Stochastic blockmodel (SBM) [1] is a popular tool for network modeling and analysis. By statistical inference, it automatically partitions the vertices in a network into disjoint groups and models the edge linkages within and between groups. It is widely used in many data mining tasks, such as connectivity pattern discovery [1], [2], community detection [3], [4], link prediction [5], [6], etc. As a fundamental model for network data, it also serves as a building block of more sophisticated models for advanced tasks, e.g., attributed graph clustering [7], and dynamic network modeling [8].

Despite its popularity in data mining and machine learning, efficient inference algorithms for SBM are rather lacking. This greatly hampers the deployment of this model for analyzing the fast-growing networks nowadays. The state-of-the-art inference method is the variational inference [9], [10]. It uses coordinate ascent [11] to optimize the variational parameters and iteratively updates them one by one in a synchronous manner. Since the number of variational parameters grows with the network size, the sequential update hinders the inference of SBM on large networks from being practical. To speed up the inference process, Daudin et al. [9] proposed to perform asynchronous update of the parameters. Although this algorithm is often efficient, it loses the convergence guarantee, which makes it hard to diagnose the convergence and may lead to poor solutions in practice.

In this paper, we propose an efficient and principled algorithm for variational inference on SBM. It replaces the costly coordinate ascent optimization with a recently proposed gradient-based optimization method called natural conjugate

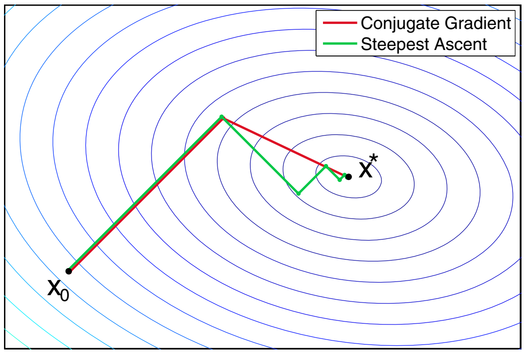


Fig. 1: Conjugate gradient method versus steepest ascent

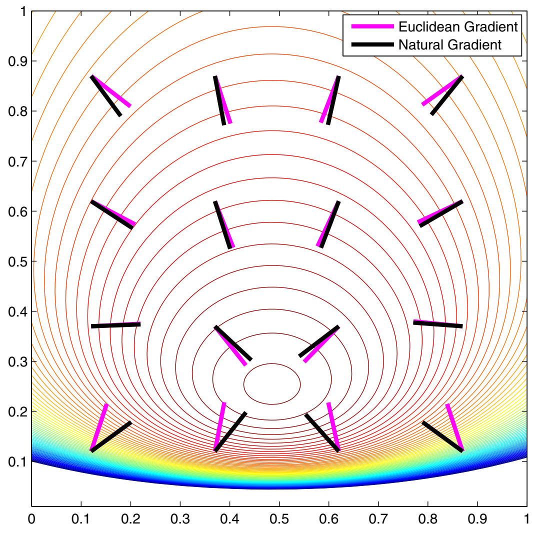


Fig. 2: Natural gradient versus Euclidean gradient

gradient (NCG) [12]. NCG leverages upon the power of both the conjugate and natural gradient. The former improves over the naive steepest ascent method [11] and finds more promising search directions by inheriting the momentum of previous directions (see Figure 1 for an illustrative example). This leads to faster convergence than steepest ascent (superlinear versus linear). The natural gradient method [13] performs the optimization in the Riemannian space, instead of in the usual Euclidean space. The family of probability distributions constitutes a Riemannian manifold [14]. Thus the space of variational parameters actually carries the Riemannian geometric information. By making use of the Riemannian nature of variational parameters, the natural gradient method finds more promising search directions than those found in the Euclidean space (See Figure 2 for an illustrative example). The coupling of the conjugate and natural gradient gives rise to more efficient optimization with convergence guarantee.

We give a succinct derivation of the natural conjugate gradient for SBM inference, based on which we develop a principled algorithm called NCG-VB. To evaluate the performance of our algorithm, we apply it to the task of community detection and conduct an extensive empirical study. Our experiments on five real-world networks show that NCG-VB runs significantly faster than state-of-the-art variational inference algorithms without sacrificing much the quality of solutions. In particular, it is up to two orders of magnitude faster than the variational Bayesian inference with synchronous update of parameters, while attaining similar or even better community detection quality. It is also about an order of magnitude faster than the variational Bayesian inference with asynchronous update of parameters, but often achieving dramatically better solution quality. We also compare NCG-VB with two gradient-based optimization methods in the Euclidean space and find that it is a clear winner in both efficiency and quality. One of the gradient-based baselines is the variant of our algorithm with conjugate gradient only. The significantly better performance of NCG-VB justifies the necessity of performing the optimization in the Riemannian space by natural gradient.

Organization. The rest of the paper is organized as follows. In Section II, we briefly review the basics of SBM and Riemannian geometry. In Section III, we derive a natural conjugate gradient method for SBM inference. We then describe the NCG-VB algorithm in Section IV and report its empirical performance in Section V, followed by a discussion of related work in Section VI. Finally, we conclude this paper in Section VII.

II. Preliminaries

In this section, we review the stochastic blockmodel, the state-of-the-art variational Bayesian method for making inference with this model, and the basics of Riemannian geometry.

A. Stochastic Blockmodel

Stochastic blockmodel (SBM) is extensively used in modeling and analyzing graph data. Consider a graph[1] $\mathcal{G} = (V, E)$ where $V = \{v_1, \ldots, v_N\}$ represents the set of N vertices and $E \subset V \times V$ represents the set of edges. SBM posits that (1) each vertex v_i belongs to one of K latent and disjoint groups, and (2) the probability that there is an edge between vertices v_i and v_j depends only on their group labels. Let $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_N)$ denote the group labels of the vertices with $Z_i \in \{1, 2, \ldots, K\}$, and $\mathbf{X} = (X_{ij})_{N\times N}$ denote the adjacency matrix of $\mathcal{G}$. Formally, SBM assumes that $\mathbf{Z}$ and $\mathbf{X}$ are generated from the distributions below:

$$\begin{aligned} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\pi}, \boldsymbol{\phi}) &= \prod_{i=1}^{N} p(Z_i|\boldsymbol{\pi}) \prod_{\substack{i,j=1 \\ i<j}}^{N} p(X_{ij}|Z_i, Z_j, \boldsymbol{\phi}), && (1) \\ Z_i &\sim \text{Multinomial}(\boldsymbol{\pi}), && (2) \\ X_{ij} &\sim \text{Bernoulli}(\phi_{Z_i Z_j}), && (3) \end{aligned}$$

where the model parameter $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_K)^T$ denotes the proportions of the K vertex groups and $\boldsymbol{\phi} = (\phi_{kl})_{K\times K}$ denotes the edge occurrence probabilities within or between vertex groups[2].

B. Variational Bayesian Inference

Given the observed adjacency matrix $\mathbf{X}$ of the graph, the main inference task of SBM is to estimate the most probable configuration of the latent group labels $\mathbf{Z}$. Variational Bayesian (VB) inference [15] tackles this problem by taking a Bayesian approach. It treats the model parameters $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ as random variables, and places prior distributions $p(\boldsymbol{\pi})$ and $p(\boldsymbol{\phi})$ on them. It then calculates the posterior distribution of $\mathbf{Z}$, $\boldsymbol{\pi}$, $\boldsymbol{\phi}$ conditioning on $\mathbf{X}$ using Bayes rule

$$p(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\mathbf{X}) \propto p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\pi}, \boldsymbol{\phi}) p(\boldsymbol{\pi}) p(\boldsymbol{\phi}),$$

and infers the most probable configuration based on the posterior,

$$\begin{aligned} \mathbf{Z}^\star &= \arg\max_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}), \\ &= \arg\max_{\mathbf{Z}} \iint p(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\mathbf{X}) \mathrm{d}\boldsymbol{\pi} \mathrm{d}\boldsymbol{\phi}. && (4) \end{aligned}$$

For mathematical convenience, conjugate priors are used for $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$,

$$\begin{aligned} \boldsymbol{\pi} &\sim \text{Dirichlet}(\boldsymbol{\alpha}), && (5) \\ p(\boldsymbol{\phi}) &= \prod_{\substack{k,l=1 \\ k\le l}}^{K} p(\phi_{kl}), && (6) \\ \phi_{kl} &\sim \text{Beta}(\boldsymbol{\beta}), && (7) \end{aligned}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the predefined hyper-parameters. Furthermore, to tackle the intractable integration in Equation 4, VB confines itself to the family of *variational distributions* that factorizes as

$$\mathcal{Q} = \left\{ q : q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}) = q(\boldsymbol{\pi}) q(\boldsymbol{\phi}) \prod_i q(Z_i) \right\},$$

and approximates the true posterior $p(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\mathbf{X})$ with the optimal variational distribution $q^\star$ that minimizes the Kullback-Leibler (KL) divergence

$$q^\star = \arg\min_{q\in\mathcal{Q}} \text{KL}\Big(q || p(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\mathbf{X})\Big). \quad (8)$$

Define

$$\mathcal{L}(q) = \sum_{\mathbf{Z}} \iint q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}) \log \frac{p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\phi})}{q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\phi})} \mathrm{d}\boldsymbol{\pi} \mathrm{d}\boldsymbol{\phi}. \quad (9)$$

[1]For the simplicity of the presentation, we consider only undirected unweighted graphs in this paper. It should be noted that SBM and our inference method can be easily generalized to the case of directed/weighted graphs.

[2]Note that $\phi_{Z_i Z_j} = \phi_{Z_j Z_i}$ for undirected graphs.

Note that

$$\mathcal{L}(q) + \mathrm{KL}\Big(q||p(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}|\mathbf{X})\Big) = \log p(\mathbf{X}),$$

whereas the RHS is the loglikelihood of the observed adjacency matrix $\mathbf{X}$ and is a constant. Since the KL divergence is non-negative by definition, $\mathcal{L}(q)$ constitutes a lower bound on the marginal loglikelihood $\log p(\mathbf{X})$, and the minimization problem in Equation 8 can be equivalently solved by maximizing this lower bound,

$$q^\star = \arg\max_{q \in \mathcal{Q}} \mathcal{L}(q).$$

VB maximizes $\mathcal{L}(q)$ using coordinate ascent. It optimizes the variational distributions $q(\boldsymbol{\pi})$, $q(\boldsymbol{\phi})$, $q(\mathbf{Z})$ in a round robin fashion [15]. In each iteration, one of the distributions is updated to improve $\mathcal{L}(q)$ with all the others fixed, and this process repeats until convergence. Specifically, the variational distributions are alternately updated [10] according to

$$\begin{aligned} q(\boldsymbol{\pi}) &\leftarrow \mathrm{Dirichlet}(\boldsymbol{\pi}|\tilde{\boldsymbol{\alpha}}), & (10)\\ q(\boldsymbol{\phi}) &\leftarrow \prod_{k \le l} \mathrm{Beta}(\phi_{kl}|\tilde{\boldsymbol{\beta}}_{kl}), & (11)\\ q(Z_i) &\leftarrow \mathrm{Multinomial}(Z_i|\tilde{\boldsymbol{\pi}}_i), \quad i \in \{1, 2, \ldots, N\} & (12)\end{aligned}$$

where $\tilde{\boldsymbol{\alpha}}$, $\tilde{\boldsymbol{\beta}}_{kl}$, $\tilde{\boldsymbol{\pi}}_i$ are the parameters of the variational distributions. They are represented as column vectors and updated as follows:

$$\begin{aligned} \tilde{\boldsymbol{\alpha}} &\leftarrow \boldsymbol{\alpha} + \tilde{\boldsymbol{\Pi}}^T \mathbf{1}, & (13)\\ \tilde{\boldsymbol{\beta}}_{kl} &\leftarrow \boldsymbol{\beta} + \frac{2 - \delta_{kl}}{2}\left(\tilde{\boldsymbol{\Pi}}_{\cdot k}^T \mathbf{X} \tilde{\boldsymbol{\Pi}}_{\cdot l}, \tilde{\boldsymbol{\Pi}}_{\cdot k}^T \bar{\mathbf{X}} \tilde{\boldsymbol{\Pi}}_{\cdot l}\right)^T, & (14)\\ \tilde{\pi}_{ik} &\propto \exp\Big\{\mathrm{E}_q\left[\log \pi_k\right] \\ &\quad + \sum_l \sum_{j \ne i} \tilde{\pi}_{jl} \mathrm{E}_q\left[\log p(X_{ij}|\phi_{kl})\right]\Big\}. & (15)\end{aligned}$$

Here, the expectations $\mathrm{E}_q[\cdot]$ are taken w.r.t. the variational distribution q and

$$\begin{aligned} \tilde{\boldsymbol{\Pi}} &\triangleq (\tilde{\pi}_{ik})_{N \times K}, \\ \bar{\mathbf{X}} &\triangleq \mathbf{1}\mathbf{1}^T - \mathbf{I} - \mathbf{X}, \end{aligned}$$

$\tilde{\boldsymbol{\Pi}}_{\cdot k}$ denotes the k-th column of $\tilde{\boldsymbol{\Pi}}$, and δ_{kl} the Kronecker delta with $\delta_{kl} = 1$ if $k = l$ and $\delta_{kl} = 0$ otherwise. $\mathbf{1}$ and $\mathbf{I}$ denote the appropriately sized all-one column vector and identity matrix, respectively.

Once the optimal variational distribution $q^\star(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\phi})$ is obtained, VB approximates the most probable group labels according to Equation 4 as follows:

$$\begin{aligned} \mathbf{Z}^\star &\approx \arg\max_{\mathbf{Z}} \iint q^\star(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\phi}) \mathrm{d}\boldsymbol{\pi} \mathrm{d}\boldsymbol{\phi} \\ &= \arg\max_{\mathbf{Z}} q^\star(\mathbf{Z}) \\ &= \left(\arg\max_k \tilde{\pi}_{1k}^\star, \ldots, \arg\max_k \tilde{\pi}_{Nk}^\star\right). & (16)\end{aligned}$$

It can be seen that the computational efficiency of VB is greatly hampered by the updating of $q(\mathbf{Z})$, or equivalently, $\tilde{\boldsymbol{\Pi}}$. Note that the parameters $\tilde{\boldsymbol{\pi}}_i$ for different vertex v_i are coupled through Equation 15. In each VB iteration, they have to be sequentially updated for every vertex in a synchronous manner. This fact renders VB computationally expensive and unscalable to large graphs with many vertices. There has been some recent work on accelerating VB using asynchronous update [9], [10]. However, the resultant algorithm loses the convergence guarantee, making it hard to diagnose the convergence in practice.

C. Riemannian Geometry

Let $\mathcal{M}$ be an m-dimensional Riemmanian manifold, and $T_{\boldsymbol{\theta}}$ the tangent space at a point $\boldsymbol{\theta} \in \mathcal{M}$. Intuitively, the tangent space $T_{\boldsymbol{\theta}}$ is the Euclidean vector space that locally linearizes $\mathcal{M}$ around $\boldsymbol{\theta}$. The inner product[3] $\langle \cdot, \cdot \rangle_{\boldsymbol{\theta}}$ is defined on $T_{\boldsymbol{\theta}} \times T_{\boldsymbol{\theta}}$ point-wise,

$$\langle \mathbf{g}_1, \mathbf{g}_2 \rangle_{\boldsymbol{\theta}} = \mathbf{g}_1^T G(\boldsymbol{\theta}) \mathbf{g}_2,$$

where $\mathbf{g}_1, \mathbf{g}_2 \in T_{\boldsymbol{\theta}}$ and $G(\boldsymbol{\theta}) = \Big(G_{ij}(\boldsymbol{\theta})\Big)$ is the positive definite Riemmanian metric tensor at $\boldsymbol{\theta}$. The length of a tangent vector $\mathbf{g} \in T_{\boldsymbol{\theta}}$ is defined by $\|\mathbf{g}\|_{\boldsymbol{\theta}} = \sqrt{\langle \mathbf{g}, \mathbf{g} \rangle_{\boldsymbol{\theta}}} = \sqrt{\mathbf{g}^T G(\boldsymbol{\theta}) \mathbf{g}}$.

A parametric family of probability distributions constitutes a Riemannian manifold, called a *statistical manifold* [14]. For a manifold $\{p(\mathbf{X}|\boldsymbol{\theta}) : \boldsymbol{\theta}\}$ with parameter $\boldsymbol{\theta}$, its Riemannian metric is given by the Fisher information matrix

$$G(\boldsymbol{\theta}) = \mathcal{I}(\boldsymbol{\theta}) = -E[\nabla_{\boldsymbol{\theta}}^2 \log p(\mathbf{X}|\boldsymbol{\theta})]$$

under mild regularity assumptions [16].

In a Riemannian space $\mathcal{M}$, the steepest ascent direction of a function $f(\boldsymbol{\theta})$ defined on $\mathcal{M}$ is given by the natural gradient

$$\tilde{\nabla}_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = G^{-1}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}).$$

III. Natural Conjugate Gradient Variational Bayes

We now propose a principled fast inference method for SBM called *natural conjugate gradient variational Bayes* (NCG-VB). The main idea is to substitute the costly coordinate ascent optimization of $\tilde{\boldsymbol{\Pi}}$ in VB (Equation 15) with a more efficient gradient method. Rather than performing the expensive sequential updating of $\tilde{\boldsymbol{\pi}}_i$ for every vertex v_i, the gradient method directly searches for promising directions in the space of $\tilde{\boldsymbol{\Pi}}$ to improve all $\tilde{\boldsymbol{\pi}}_i$'s simultaneously. In particular, our method builds upon the conjugate gradient (CG) optimization, which enjoys fast convergence (superlinear versus linear convergence rate of steepest ascent). Furthermore, it performs the optimization on the Riemannian manifold instead of the Euclidean space. As aforementioned, this could significantly accelerate the optimization process.

In the following, we first develop the Euclidean space CG method for optimizing $\tilde{\boldsymbol{\Pi}}$. We then describe how to switch to the Riemannian space and derive the natural gradient for this purpose.

A. *Optimizing $\tilde{\boldsymbol{\Pi}}$ with CG*

Given an unconstrained maximization problem with objective function $f(\boldsymbol{\theta}) : \mathcal{R}^n \to \mathcal{R}$, CG [16] starts from an initial

[3]This inner product generalizes the dot product in the Euclidean space $\mathcal{R}^m$. That is, $\mathcal{R}^m$ can be subsumed into Riemannian manifolds, and the associated Riemmanian metric is the identity function, i.e., $G_{ij}(\boldsymbol{\theta}) = \delta_{ij}$.

solution $\boldsymbol{\theta}^{(0)} \in \mathcal{R}^n$ and iteratively optimizes it as follows

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} + \lambda^{(t)}\mathbf{d}^{(t)}. \tag{17}$$

Here, $\mathbf{d}^{(t)}$ is the conjugate gradient at the t-th step [16] defined as

$$\mathbf{d}^{(t)} = \begin{cases} \mathbf{g}^{(t)}, & t = 0 \\ \mathbf{g}^{(t)} + \frac{\|\mathbf{g}^{(t)}\|^2}{\|\mathbf{g}^{(t-1)}\|^2}\mathbf{d}^{(t-1)}, & t > 0 \end{cases}$$

where $\mathbf{g}^{(t)} = \nabla f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}$. $\lambda^{(t)}$ is the step size which can be set to a small constant or adjusted adaptively, e.g. by line search [11].

Recall that our optimization problem is to maximize the variational bound $\mathcal{L}(q)$ with respect to $\tilde{\mathbf{\Pi}}$. Note that $\tilde{\mathbf{\Pi}}$ consists of the parameters of N multinomial distributions. Therefore, the optimization problem is subject to the constraint $\tilde{\mathbf{\Pi}}\mathbf{1} = \mathbf{1}$. In order to apply CG, we need to transform the optimization problem into an unconstrained one, and we do so by re-parametrization.

We start by simplifying the objective function $\mathcal{L}(q)$. Plugging the parametric forms of $p(\cdot)$ (Equations 1–3, 5–7) and $q(\cdot)$ (Equations 10–15) into Equation 9, we obtain an equivalent objective function over the variational parameters:

$$\begin{aligned} \mathcal{L}(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\beta}}, \tilde{\mathbf{\Pi}}) \quad = \quad & \sum_k (\alpha_k + \tilde{\mathbf{\Pi}}_{\cdot k}^T \mathbf{1} - \tilde{\alpha}_k) \mathrm{E}_q[\log \pi_k] \\ & + \sum_{k \le l} [\beta_1 + \frac{2-\delta_{kl}}{2} \tilde{\mathbf{\Pi}}_{\cdot k}^T \mathbf{X} \tilde{\mathbf{\Pi}}_{\cdot l} - \tilde{\beta}_{kl,1}] \mathrm{E}_q[\log \phi_{kl}] \\ & + \sum_{k \le l} [\beta_2 + \frac{2-\delta_{kl}}{2} \tilde{\mathbf{\Pi}}_{\cdot k}^T \bar{\mathbf{X}} \tilde{\mathbf{\Pi}}_{\cdot l} - \tilde{\beta}_{kl,2}] \mathrm{E}_q[\log(1-\phi_{kl})] \\ & + \mathcal{A}(\tilde{\mathbf{\Pi}}) + \log \frac{\mathcal{B}(\tilde{\boldsymbol{\alpha}})}{\mathcal{B}(\boldsymbol{\alpha})} + \sum_{k \le l} \log \frac{\mathcal{B}(\tilde{\boldsymbol{\beta}}_{kl})}{\mathcal{B}(\boldsymbol{\beta})}, \end{aligned} \tag{18}$$

where

$$\begin{aligned} \mathcal{B}(\boldsymbol{\alpha}) \quad &= \quad \frac{\prod_{k=1}^K \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^K \alpha_k)}, \\ \mathcal{A}(\tilde{\mathbf{\Pi}}) \quad &\triangleq \quad -\sum_{i,k} \tilde{\pi}_{ik} \log \tilde{\pi}_{ik}. \end{aligned}$$

Note that the optima of $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\beta}}$ are characterized by Equations 13 and 14, and are both functions of $\tilde{\mathbf{\Pi}}$. Denote these functions by $\tilde{\boldsymbol{\alpha}}^\star(\tilde{\mathbf{\Pi}})$ and $\tilde{\boldsymbol{\beta}}^\star(\tilde{\mathbf{\Pi}})$, respectively. Plugging them back into Equation 18, we can see that to maximize $\mathcal{L}(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\beta}}, \tilde{\mathbf{\Pi}})$ is equivalent to maximize the following objective function over $\tilde{\mathbf{\Pi}}$ only:

$$\mathcal{L}(\tilde{\mathbf{\Pi}}) = \mathcal{A}(\tilde{\mathbf{\Pi}}) + \log \frac{\mathcal{B}(\tilde{\boldsymbol{\alpha}}^\star(\tilde{\mathbf{\Pi}}))}{\mathcal{B}(\boldsymbol{\alpha})} + \sum_{k \le l} \log \frac{\mathcal{B}(\tilde{\boldsymbol{\beta}}_{kl}^\star(\tilde{\mathbf{\Pi}}))}{\mathcal{B}(\boldsymbol{\beta})}. \tag{19}$$

To remove the constraint on $\tilde{\mathbf{\Pi}}$, we re-parameterize the multinomial distribution $q(Z_i|\tilde{\boldsymbol{\pi}}_i)$ for all $i \in \{1, 2, \ldots, N\}$ using the minimal exponential family representation [17] as follows:

$$q(Z_i|\boldsymbol{\theta}_i) = \exp\left\{\theta_i^T \mathbf{Z}_i^* - A(\boldsymbol{\theta}_i)\right\}.$$

Here[4], $\mathbf{Z}_i^* = (\delta_{Z_i,1}, \delta_{Z_i,2}, \cdots, \delta_{Z_i,K-1})^T$. $\boldsymbol{\theta}_i$ is the natural parameter and relates to $\tilde{\boldsymbol{\pi}}_i$ through

$$\tilde{\pi}_{ik}(\boldsymbol{\theta}_i) \quad = \quad \begin{cases} \exp\{\theta_{ik} - A(\boldsymbol{\theta}_i)\}, & k = 1, \cdots, K-1 \\ \exp\{-A(\boldsymbol{\theta}_i)\}, & k = K \end{cases} \tag{20}$$

and $A(\boldsymbol{\theta}_i) = \log(1 + \sum_{k=1}^{K-1} \exp\{\theta_{ik}\})$. Note that there is no constraint on $\boldsymbol{\theta}_i$. Let $\mathbf{\Theta} = (\theta_{ik})_{N \times (K-1)}$ and

$$S(\mathbf{\Theta}) \triangleq \mathcal{L}\left(\tilde{\mathbf{\Pi}}(\mathbf{\Theta})\right).$$

We can thus reduce the constrained optimization of $\mathcal{L}(\tilde{\mathbf{\Pi}})$ to the unconstrained optimization $\max_{\mathbf{\Theta}} S(\mathbf{\Theta})$, to which CG can be readily applied.

B. Natural Conjugate Gradient

CG searches for good directions in the Euclidean space to optimize $S(\mathbf{\Theta})$. We note that, however, the variable $\mathbf{\Theta}$ is the parameter of the family of distributions $q(\mathbf{Z}|\mathbf{\Theta})$, which constitutes a Riemannian manifold. Therefore, optimizing $S(\mathbf{\Theta})$ in the Riemannian space could give rise to better directions for improving $S(\mathbf{\Theta})$ and hence faster convergence. Based on this idea, we now develop a natural conjugate gradient (NCG) method for optimizing $S(\mathbf{\Theta})$.

NCG builds upon the framework of the CG method but substitutes all the gradients and vector lengths derived in the Euclidean space with their Riemannian counterparts [12]. Specifically, it re-defines the conjugate gradient in the Riemannian space as follows:

$$\tilde{\mathbf{d}}^{(t)} = \begin{cases} \tilde{\mathbf{g}}^{(t)}, & t = 0 \\ \tilde{\mathbf{g}}^{(t)} + \frac{\|\tilde{\mathbf{g}}^{(t)}\|^2_{\mathbf{\Theta}^{(t)}}}{\|\tilde{\mathbf{g}}^{(t-1)}\|^2_{\mathbf{\Theta}^{(t-1)}}}\tilde{\mathbf{d}}^{(t-1)}, & t > 0 \end{cases} \tag{21}$$

where $\tilde{\mathbf{g}}^{(t)} = \tilde{\nabla}_{\mathbf{\Theta}} S(\mathbf{\Theta})|_{\mathbf{\Theta}=\mathbf{\Theta}^{(t)}}$ is the natural gradient of $S(\mathbf{\Theta})$. Called the *natural conjugate gradient*, $\tilde{\mathbf{d}}^{(t)}$ will serve as the direction to improve the objective function $S(\mathbf{\Theta})$ in each iteration. This method has been shown to retain the superlinear convergence rate [12].

We are left with deriving the natural gradient $\tilde{\mathbf{g}}$ and its square length $\|\tilde{\mathbf{g}}\|^2_{\mathbf{\Theta}}$. Both of them are based on the Riemannian metric $G(\mathbf{\Theta})$. Since $q(\mathbf{Z}|\mathbf{\Theta})$ factorizes, $G(\mathbf{\Theta})$ can be simplified as

$$\begin{aligned} G(\mathbf{\Theta}) \quad &= \quad \mathrm{diag}\Big(\mathcal{I}(\boldsymbol{\theta}_1), \ldots, \mathcal{I}(\boldsymbol{\theta}_N)\Big) \\ &= \quad \mathrm{diag}\Big(\nabla^2_{\boldsymbol{\theta}_1} A(\boldsymbol{\theta}_1), \ldots, \nabla^2_{\boldsymbol{\theta}_N} A(\boldsymbol{\theta}_N)\Big) \\ &= \quad \mathrm{diag}\Big(\nabla_{\boldsymbol{\theta}_1} \tilde{\mathbf{\Pi}}^*_{1\cdot}, \ldots, \nabla_{\boldsymbol{\theta}_N} \tilde{\mathbf{\Pi}}^*_{N\cdot}\Big) \\ &= \quad \nabla_{\mathbf{\Theta}} \tilde{\mathbf{\Pi}}^*, \end{aligned} \tag{22}$$

where $\tilde{\mathbf{\Pi}}^* = (\tilde{\pi}_{ik})_{N \times (K-1)}$ and $\tilde{\mathbf{\Pi}}^*_{i\cdot}$ denotes the i-th row of $\tilde{\mathbf{\Pi}}^*$, and the second and third equalities follow from the properties of exponential family [17]. The natural gradient can thus be derived as follows

$$\begin{aligned} \tilde{\mathbf{g}} \quad &= \quad \tilde{\nabla}_{\mathbf{\Theta}} S(\mathbf{\Theta}) \\ &= \quad G^{-1}(\mathbf{\Theta}) \nabla_{\mathbf{\Theta}} S(\mathbf{\Theta}) \\ &= \quad G^{-1}(\mathbf{\Theta}) \left(\nabla_{\mathbf{\Theta}} \tilde{\mathbf{\Pi}}^*\right)^T \nabla_{\tilde{\mathbf{\Pi}}^*} \mathcal{L}(\tilde{\mathbf{\Pi}}), \end{aligned}$$

[4]We use the superscript $*$ to indicate a dimension reduced counterpart and $\star$ to indicate an optimal value.

where the last equality follows due to the chain rule of gradient. Note that $\nabla_{\mathbf{\Theta}}\tilde{\mathbf{\Pi}}^*$ is symmetric. Plugging Equation 22, we obtain that

$$\tilde{\mathbf{g}} = \nabla_{\tilde{\mathbf{\Pi}}^*}\mathcal{L}(\tilde{\mathbf{\Pi}}) = \begin{pmatrix} (\mathbf{I}, -\mathbf{1})\nabla_{\tilde{\boldsymbol{\pi}}_1}\mathcal{L}(\tilde{\mathbf{\Pi}}) \\ \vdots \\ (\mathbf{I}, -\mathbf{1})\nabla_{\tilde{\boldsymbol{\pi}}_N}\mathcal{L}(\tilde{\mathbf{\Pi}}) \end{pmatrix}, \qquad (23)$$

where the second equality follows from

$$\tilde{\boldsymbol{\pi}}_{iK} = 1 - \sum_{k=1}^{K-1} \tilde{\boldsymbol{\pi}}_{ik}$$

and

$$\nabla_{\tilde{\mathbf{\Pi}}^*_{i\cdot}}\mathcal{L}(\tilde{\mathbf{\Pi}}) = (\mathbf{I}, -\mathbf{1})\nabla_{\tilde{\boldsymbol{\pi}}_i}\mathcal{L}(\tilde{\mathbf{\Pi}}).$$

Consequently, we can derive the square length of $\tilde{\mathbf{g}}$ as follows

$$\begin{aligned} \|\mathbf{g}\|^2_{\mathbf{\Theta}} &= \mathbf{g}^T G(\mathbf{\Theta})\mathbf{g} \\ &= [\nabla_{\tilde{\mathbf{\Pi}}^*}\mathcal{L}(\tilde{\mathbf{\Pi}})]^T(\nabla_{\mathbf{\Theta}}\tilde{\mathbf{\Pi}}^*)[\nabla_{\tilde{\mathbf{\Pi}}^*}\mathcal{L}(\tilde{\mathbf{\Pi}})] \\ &= \sum_{i=1}^{N}[\nabla_{\tilde{\boldsymbol{\pi}}_i}\mathcal{L}(\tilde{\mathbf{\Pi}})]^T[\mathrm{diag}(\tilde{\boldsymbol{\pi}}_i) - \tilde{\boldsymbol{\pi}}_i^T\tilde{\boldsymbol{\pi}}_i][\nabla_{\tilde{\boldsymbol{\pi}}_i}\mathcal{L}(\tilde{\mathbf{\Pi}})] \end{aligned} \qquad (24)$$

where we have used that

$$\nabla_{\boldsymbol{\theta}_i}\tilde{\mathbf{\Pi}}^*_{i\cdot} = \mathrm{diag}(\tilde{\mathbf{\Pi}}^*_{i\cdot}) - (\tilde{\mathbf{\Pi}}^*_{i\cdot})^T(\tilde{\mathbf{\Pi}}^*_{i\cdot})$$

by Equation 20 and

$$(\mathbf{I}, -\mathbf{1})^T(\nabla_{\boldsymbol{\theta}_i}\tilde{\mathbf{\Pi}}^*_{i\cdot})(\mathbf{I}, -\mathbf{1}) = \mathrm{diag}(\tilde{\boldsymbol{\pi}}_i) - \tilde{\boldsymbol{\pi}}_i\tilde{\boldsymbol{\pi}}_i^T.$$

Note that, while the natural gradient and its squared length are defined with respect to $\mathbf{\Theta}$, they can be computed based on $\tilde{\mathbf{\Pi}}$ and the costly matrix inverse $G^{-1}(\mathbf{\Theta})$ is avoided. In particular, they depend on the gradient $\nabla_{\tilde{\boldsymbol{\pi}}_i}\mathcal{L}(\tilde{\mathbf{\Pi}})$, which can be computed as follows

$$\begin{aligned} \frac{\partial\mathcal{L}(\tilde{\mathbf{\Pi}})}{\partial\tilde{\pi}_{ik}} &= \sum_{l=1}^{K}(\mathbf{X}_{i\cdot}\tilde{\mathbf{\Pi}}_{\cdot l}, \bar{\mathbf{X}}_{i\cdot}\tilde{\mathbf{\Pi}}_{\cdot l})\nabla\log\mathcal{B}(\tilde{\boldsymbol{\beta}}^\star_{kl}(\tilde{\mathbf{\Pi}})) \\ &\quad +\psi(\tilde{\alpha}^\star_k(\tilde{\mathbf{\Pi}})) - \log\tilde{\pi}_{ik} - 1. \end{aligned} \qquad (25)$$

IV. The Algorithm

The pseudo code of our NCG-VB method for SBM is given in Algorithm 1. Given a graph with adjacency matrix $\mathbf{X}$ and a group number K, the algorithm starts from an initial value of the model parameter $\mathbf{\Theta}$, searches for the optimal parameter, based on which it then outputs the best group assignment $\mathbf{Z}^\star$ of the vertices.

The algorithm alternates between the two equivalent parameter representations $\tilde{\mathbf{\Pi}}$ and $\mathbf{\Theta}$. In each iteration (the while loop since line 4), it first works in the space of $\tilde{\mathbf{\Pi}}$ and optimizes $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\beta}}$ (lines 5–6). It then evaluates the objective function (line 7) and checks the convergence conditions (lines 8–9). In particular, the algorithm terminates once the relative increase in the objective function is less than the predefined tolerance η or the number of iterations exceeds the limit $t_{\max}$. Finally, it switches to the space of $\mathbf{\Theta}$ and optimizes this parameter using NCG (lines 10–17).

Algorithm 1 NCG-VB

```
Input: adjacency matrix $\mathbf{X}$, group number $K$, initial value $\mathbf{\Theta}$,
    tolerance $\eta$, maximum number of iterations $t_{\max}$
Output: vertex group assignment $\mathbf{Z}^\star$
 1: $\mathcal{L}_{\text{old}} \leftarrow -\infty$
 2: $\lambda \leftarrow 1$
 3: $t \leftarrow 1$
 4: while true do
 5:     Calculate $\tilde{\mathbf{\Pi}}$ using Eq. 20
 6:     Update $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\beta}}$ using Eqs. 13–14
 7:     Calculate $\mathcal{L}$ using Eq. 19
 8:     if $0 \le \frac{\mathcal{L}-\mathcal{L}_{\text{old}}}{|\mathcal{L}|} < \eta$ or $t = t_{\max}$ then
 9:         break
10:     if $\mathcal{L} \ge \mathcal{L}_{\text{old}}$ then
11:         Update $\tilde{\mathbf{d}}$ using Eqs. 21, 23–25
12:         $\mathbf{\Theta}_{\text{old}} \leftarrow \mathbf{\Theta}$
13:         $\mathbf{\Theta} \leftarrow \mathbf{\Theta}_{\text{old}} + \lambda\tilde{\mathbf{d}}$
14:         $\mathcal{L}_{\text{old}} \leftarrow \mathcal{L}$
15:     else
16:         $\lambda \leftarrow \frac{\lambda}{2}$
17:         $\mathbf{\Theta} \leftarrow \mathbf{\Theta}_{\text{old}} + \lambda|\frac{\mathcal{L}-\mathcal{L}_{\text{old}}}{\mathcal{L}}|\tilde{\mathbf{d}}$
18:     $t \leftarrow t + 1$
19: return $(\arg\max_k \tilde{\pi}_{1k}, \cdots, \arg\max_k \tilde{\pi}_{Nk})$
```

It is worth mentioning our strategy of tuning the step size λ of NCG. We initialize λ at 1. In each iteration, if NCG overshoots and yields a decrease in the objective function ($\mathcal{L} < \mathcal{L}_{\text{old}}$), we shrink the step size by half (line 16), rewind to the solution $\mathbf{\Theta}_{\text{old}}$ of the previous iteration, and make a very small jump to recover the search (line 17). Otherwise, the step size remains unchanged. Despite the simplicity, this strategy works well in our experiments.

A. Variant for Community Detection

SBM captures both assortative and disassortative graph structures [18]. When performing community detection, however, we are only interested in assortative ones. This could be achieved by suppressing the inter-group edge occurrence probabilities and clamping them to a small positive constant ϵ [19], i.e., setting $\phi_{kl} = \epsilon$ for all $k \neq l$.

This constraint can be easily implemented in NCG-VB. We only need to modify the calculation of the objective function in Equation 19 and its gradient in Equation 25 as follows:

$$\begin{aligned} \mathcal{L}(\tilde{\mathbf{\Pi}}) &\triangleq \Big(\log\epsilon, \log(1-\epsilon)\Big)\sum_{k<l}(\tilde{\mathbf{\Pi}}^T_{\cdot k}\mathbf{X}\tilde{\mathbf{\Pi}}_{\cdot l}, \tilde{\mathbf{\Pi}}^T_{\cdot k}\bar{\mathbf{X}}\tilde{\mathbf{\Pi}}_{\cdot l})^T \\ &\quad +\mathcal{A}(\tilde{\mathbf{\Pi}}) + \log\frac{\mathcal{B}(\tilde{\boldsymbol{\alpha}}^\star(\tilde{\mathbf{\Pi}}))}{\mathcal{B}(\boldsymbol{\alpha})} + \sum_k \log\frac{\mathcal{B}(\tilde{\boldsymbol{\beta}}^\star_{kk}(\tilde{\mathbf{\Pi}}))}{\mathcal{B}(\boldsymbol{\beta})}, \\ \frac{\partial\mathcal{L}(\tilde{\mathbf{\Pi}})}{\partial\tilde{\pi}_{ik}} &= \Big(\log\epsilon, \log(1-\epsilon)\Big)\sum_{l\neq k}\Big(\mathbf{X}_{i\cdot}\tilde{\mathbf{\Pi}}_{\cdot l}, \bar{\mathbf{X}}_{i\cdot}\tilde{\mathbf{\Pi}}_{\cdot l}\Big)^T \\ &\quad +\Big(\mathbf{X}_{i\cdot}\tilde{\mathbf{\Pi}}_{\cdot k}, \bar{\mathbf{X}}_{i\cdot}\tilde{\mathbf{\Pi}}_{\cdot k}\Big)\nabla\log\mathcal{B}(\tilde{\boldsymbol{\beta}}^\star_{kk}(\tilde{\mathbf{\Pi}})) \\ &\quad +\psi(\tilde{\alpha}^\star_k(\tilde{\mathbf{\Pi}})) - \log\tilde{\pi}_{ik} - 1. \end{aligned}$$

The rest of the algorithm remains unchanged.

TABLE I: Datasets

Network	Number of vertices	Number of edges	K
ca-GrQc	5,242	14,484	50
ca-HepTh	9,877	25,973	100
PGP	10,680	24,316	100
twitter	81,306	1,342,296	80
wordnet	146,005	656,999	150

V. Experimental Evaluation

We now empirically evaluate the performance of NCG-VB[5] on community detection task and compare it with state-of-the-art variational inference algorithms.

A. *Datasets*

We use five real-world networks from a variety of domains:

- **ca-GrQc**: A collaboration network from the e-print arXiv in the subject of General Relativity and Quantum Cosmology. Each vertex represents an author and each edge represents a co-authorship between two authors.
- **ca-HepTh**: A collaboration network from the e-print arXiv in the subject of High Energy Physics - Theory. The semantics of vertices and edges are the same as in **ca-GrQc**.
- **PGP**: A contact network of users of the Pretty Good Privacy (PGP) algorithm. Each vertex is a user of the PGP algorithm and each edge represents an interaction between two users.
- **twitter**: A social network from Twitter. Each vertex represents a Twitter user and each edge represents the existence of friendship between two users.
- **wordnet**: A lexical network of English words. Each vertex is an English word and each edge represents the existence of a relationship (such as synonymy, antonymy, meronymy, etc.) between two words.

The datasets **ca-GrQc**, **ca-HepTh** and **twitter** are obtained from the Stanford Network Analysis Project[6], and **PGP** and **wordnet** are obtained from the Koblenz Network Collection[7]. Among them, **twitter** is directed and unweighted. We convert it to an undirected version by symmetrizing the associated adjacency matrix, i.e., $\mathbf{X} \leftarrow \max(\mathbf{X}, \mathbf{X}^T)$. All the other networks are undirected and unweighted. The statistics are summarized in Table I.

B. *Experimental Settings*

We detail the experimental settings in this subsection, including the baseline algorithms for comparison, the quality measures, and the initialization of parameters.

[5]The source code of NCG-VB can be obtained at https://github.com/zhiqiangxu2001/NCG-SBM

[6]http://snap.stanford.edu/data

[7]http://konect.uni-koblenz.de/networks/

1) Baseline Algorithms:

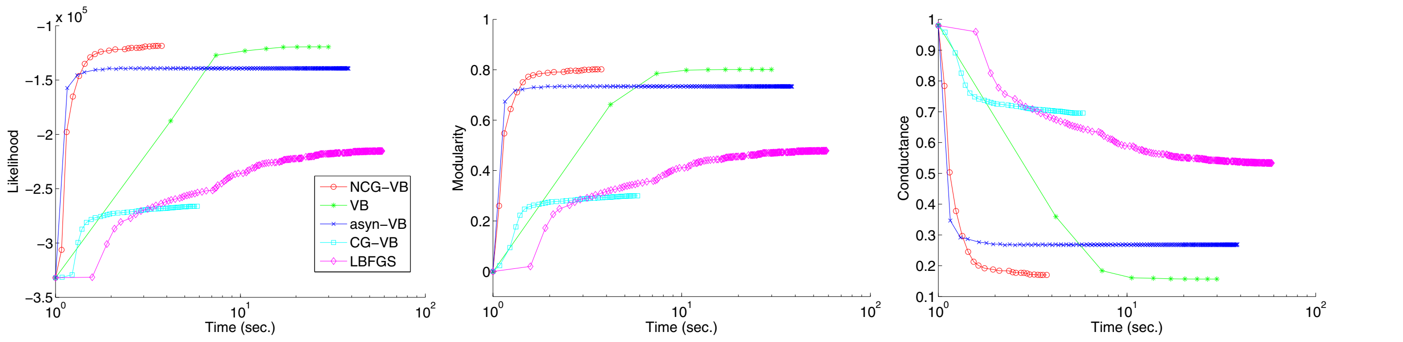
- **VB**: The vanilla variational Bayesian inference that performs synchronous update on the variational parameters $\tilde{\mathbf{\Pi}}$.
- **asyn-VB**: The variant of VB that performs asynchronous update on $\tilde{\mathbf{\Pi}}$ using the fixed-point algorithm [9], [10].
- **CG-VB**: The variant that uses solely the conjugate gradient method to optimize $\tilde{\mathbf{\Pi}}$ as described in Section III-A.
- **LBFGS**: The variant that uses the popular limited memory BFGS method [11] to optimize $\tilde{\mathbf{\Pi}}$. We use the implementation in [20].

All the algorithms were implemented in Matlab and tested on a machine with Linux OS, Intel Xeon 2.80GHz CPU, and 25GB of RAM.

2) Quality Measures: We use three different measures to assess the quality of the detected communities.

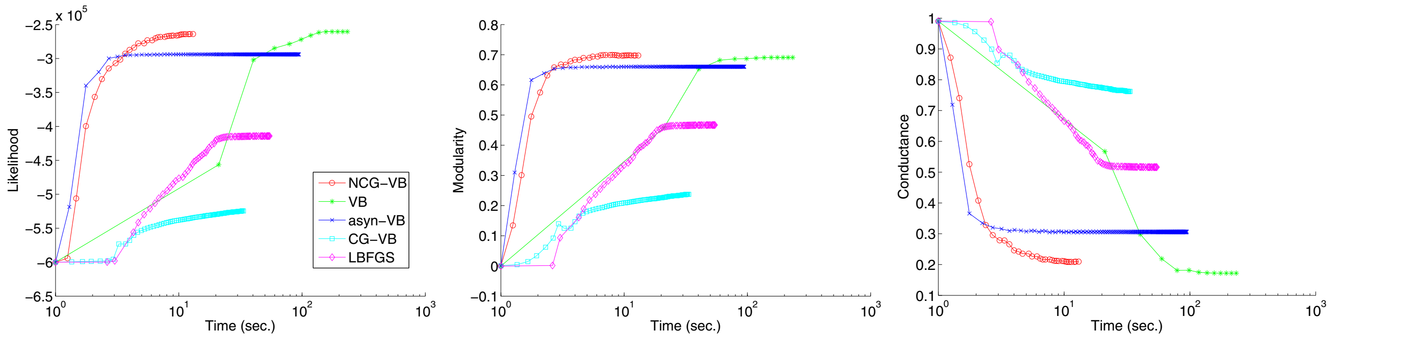
- **Likelihood Bound**: The variational lower bound on the marginal loglikelihood which our algorithm and all the baselines aim to maximize. A higher value of the bound means a better optimization performance.
- **Modularity**: A commonly used measure for evaluating the quality of a division of a network into communities [21]. It is defined as the number of intra-community edges relative to a null model of a random graph with the same degree distribution. Let m_k denote the number of edges within community k and c_k the number of edges connecting community k with the other communities. Formally, Modularity $= \sum_{k=1}^{K}[\frac{m_k}{|E|} - (\frac{2m_k+c_k}{2|E|})^2]$, where $|E|$ represents the number of edges in the network. A higher value of modularity means a better community structure detected.
- **Conductance**: Another commonly used measure to assess community quality. When used to measure the quality of a single community, it is defined as the fraction of the edges leaving the community among all the edges involving the community [22]. To measure the quality of community detection solution, we use the average conductance of all the detected communities, i.e., $\frac{1}{K}\sum_{k=1}^{K}\frac{c_k}{2m_k+c_k}$. A lower value of conductance means sparser inter-community connectivity and thus better community detection.

3) Initialization of Parameters: We initialize all the algorithms with the same value of the variational parameters. For NCG-VB and the two baselines CG-VB and LBFGS, we use the initial value of $\mathbf{\Theta} = (\theta_{ik})_{N\times(K-1)}$ sampled from the standard normal distributions as $\theta_{ik} \sim N(0, 1)$. For VB and asyn-VB, we use the same initial value of $\tilde{\mathbf{\Pi}}$ calculated from the initial value of $\mathbf{\Theta}$ using Equation 20. We set the step size of CG-VB in the same way as our algorithm, and the step size of LBFGS using line search as in [20]. For all the algorithms, we use the same cluster number K, which is chosen such that the average cluster size is about 100 for the first three datasets, and



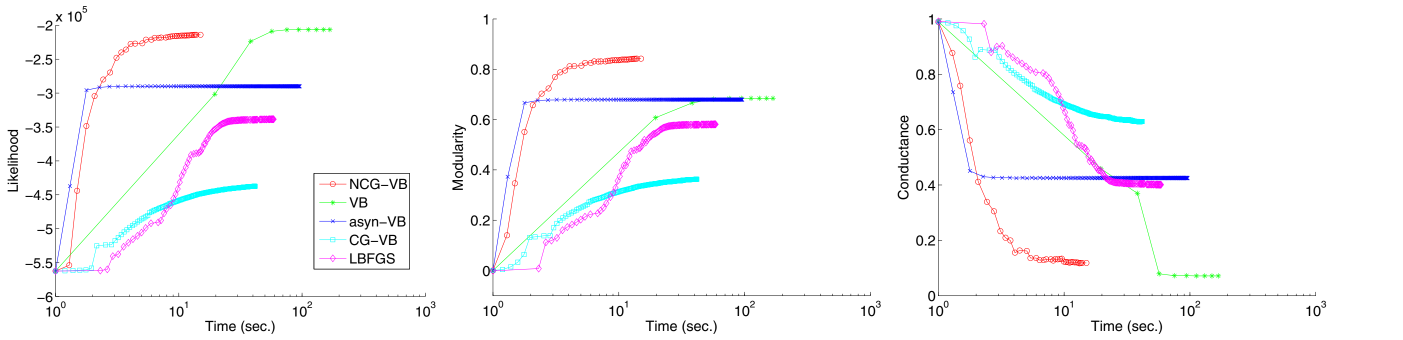
(a) Likelihood Bound (b) Modularity (c) Conductance

Fig. 3: Performance on **ca-GrQc**



(a) Likelihood Bound (b) Modularity (c) Conductance

Fig. 4: Performance on **ca-HepTh**



(a) Likelihood Bound (b) Modularity (c) Conductance

Fig. 5: Performance on **PGP**

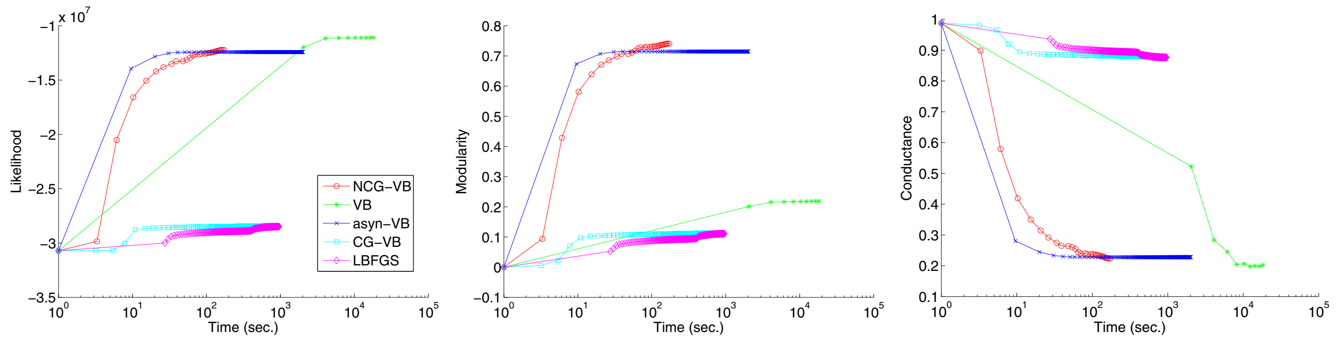
is about 1000 for the two larger datasets (see Table I). We also investigate the sensitivity of the performance when K varies. The inter-group edge probability is fixed at $\epsilon = 10^{-10}$ and the hyper-parameters are set to $\boldsymbol{\alpha} = \mathbf{1}$, $\boldsymbol{\beta} = \mathbf{1}$. The tolerance on the relative precision of likelihood bound is set to $\eta = 10^{-6}$ and the maximum iteration number is set to $t_{\max} = 200$.

We run each algorithm 10 times, each run with a different initial value of $\boldsymbol{\Theta}$ or $\tilde{\boldsymbol{\Pi}}$. We report the result of the run with the highest likelihood bound achieved by each algorithm, as well as the average performance results. We terminate an algorithm if it has run for more than 5 hours.

C. Performance Results

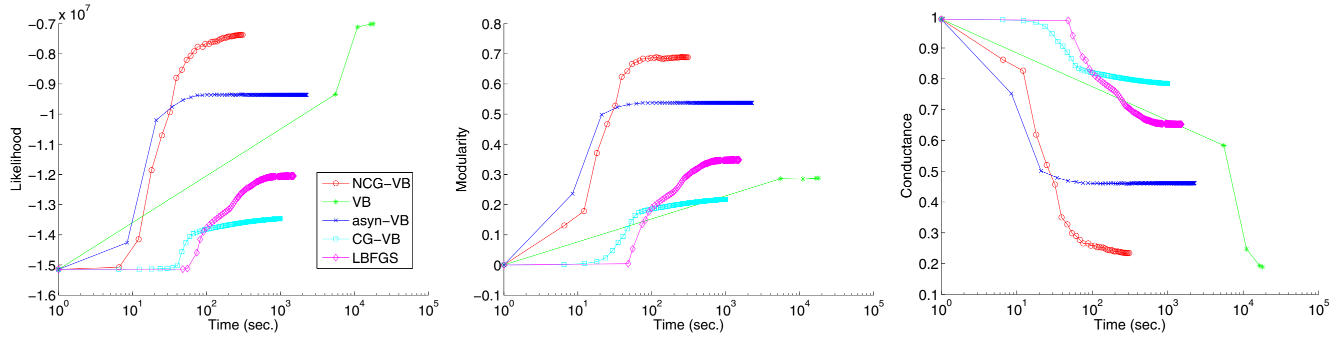
We report the performance results of the algorithms in Figures 3–7, which show the convergence curves of their likelihood bound, modularity, and conductance on each dataset. Each point on a convergence curve denotes one iteration of the algorithm.

We first compare the performance of our algorithm NCG-VB with that of VB. In terms of efficiency, NCG-VB converges



(a) Likelihood Bound (b) Modularity (c) Conductance

Fig. 6: Performance on **twitter**



(a) Likelihood Bound (b) Modularity (c) Conductance

Fig. 7: Performance on **wordnet**

much faster than VB. Specifically, NCG-VB is over an order of magnitude faster on the three small datasets. On the two large datasets **twitter** and **wordnet**, the advantage of NCG-VB is even more pronounced. It takes only minutes to converge, while VB fails to converge in 5 hours. Under this early termination, the gap is up to two orders of magnitude, which should be even larger if we drive VB to convergence.

In terms of solution quality, NCG-VB is competitive to VB. Specifically, NCG-VB achieves comparable likelihood bound and conductance to VB on all datasets. When measured by modularity, NCG-VB performs similarly to VB on the two collaboration networks **ca-GRQc** and **ca-HepTh**, but dramatically outperforms VB on the other three datasets. The gap is particularly wide on large datasets **twitter** and **wordnet**.

We next compare NCG-VB with asyn-VB. We can see that asyn-VB improves the likelihood bound slightly faster than NCG-VB at the initial few iterations. After this period, however, it often gets trapped in poor solutions with relatively low likelihood. Due to the lack of convergence guarantee, its likelihood bound keeps fluctuating (e.g., see Figure 8 for a zoomed view on **wordnet**) and never converges within the given 200 iterations. In contrast, NCG-VB converges quickly in less than 50 iterations on all the datasets, and is about 10 times faster than asyn-VB in running time. Furthermore, it achieves consistently higher quality solutions than asyn-VB in all three measures and on all the datasets.

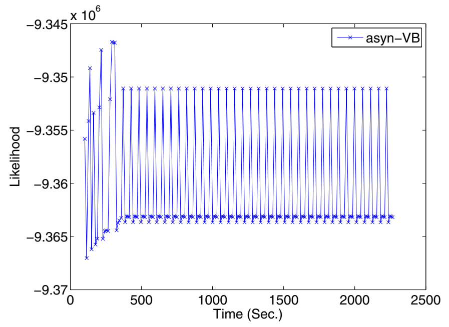
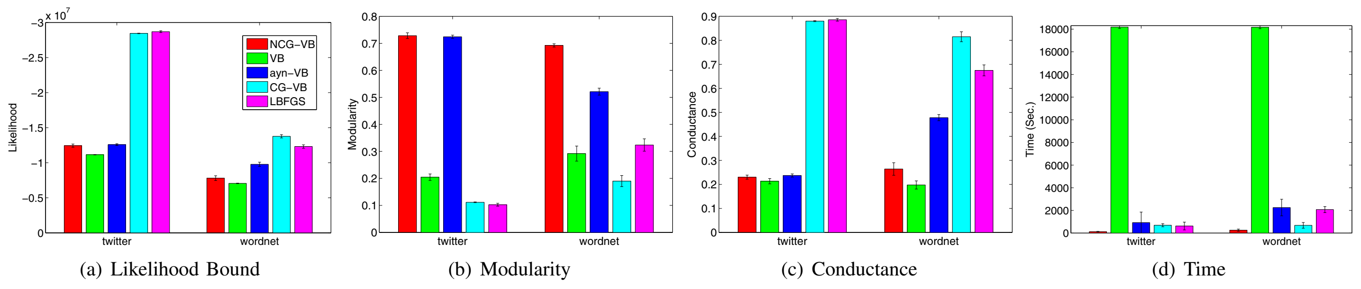


Fig. 8: The tail of the convergence curve of asyn-VB on **wordnet**.

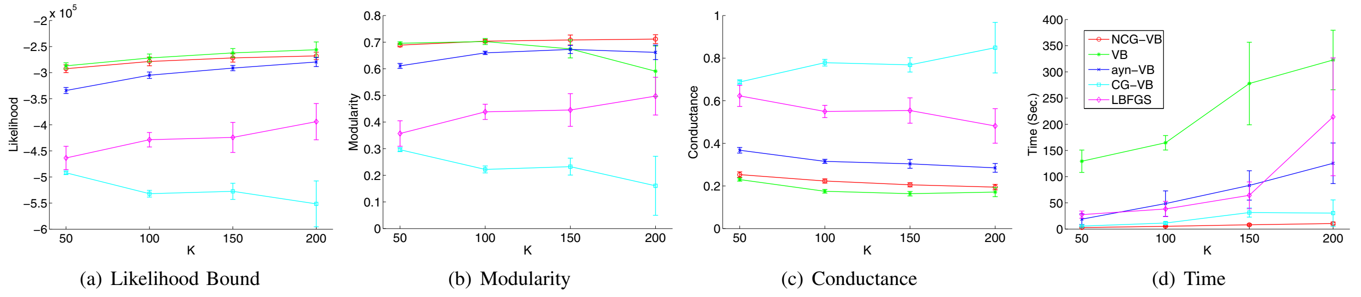
We also compare the performance of NCG-VB with that of CG-VB and LBFGS. NCG-VB consistently and significantly outperforms these alternatives in terms of both efficiency and solution quality. It is worth noting that both alternatives perform gradient-based optimization in the Euclidean space, which usually leads to small improvement in the likelihood bound per iteration. This justifies the necessity of embracing the Riemannian nature of the variational parameters and thus the need of the natural gradient method.

We now report the average performance of the algorithms



(a) Likelihood Bound (b) Modularity (c) Conductance (d) Time

Fig. 9: Average performance on **twitter** and **wordnet**



(a) Likelihood Bound (b) Modularity (c) Conductance (d) Time

Fig. 10: Varying cluster number on **ca-HepTh**.

on the two largest datasets. Figure 9 shows the average values and standard deviations of likelihood bound[8], modularity, conductance, and time cost. As we can see, the average performance is consistent with the best results reported above. The performance variation at different initial points is shown to be small. Similar conclusions can be drawn on the three small datasets. We omit the details due to the space limit.

To examine the performance of the algorithms with a varying cluster number, we report in Figure 10 the average results in terms of the four measures on the **ca-HepTh** dataset, with $K = 50, 100, 150$ and 200. It demonstrates that our algorithm NCG-VB presents the advantages over baselines similar to those in the case of $K = 100$ reported previously. The performance of NCG-VB remains relatively stable as the cluster number varies in a wide range. Stable performance of NCG-VB with different initial points is also observed. Moreover, the efficiency advantage of NCG-VB becomes more pronounced as the cluster number increases.

To summarize, our algorithm NCG-VB is significantly faster than all the baselines. It also generates solutions with competitive quality to VB, and consistently outperforms the other alternatives.

VI. Related Work

Despite the popularity of VB inference, there has been limited work on boosting it for SBM. Daudin and Robin [9] and Latouche et al. [10] proposed to replace the costly sequential optimization of the local parameters using the approximate asynchronous update. The latter can exploit the data sparsity more effectively but loses the convergence guarantee as a side effect. Recently, Gopalan et al. [19] proposed stochastic variational inference (SVI) for the mixed-membership variant of SBM and used it for detecting overlapping communities. However, the inference of mixed-membership SBM has no synchronization issue during updating variational parameters, and thus is different from our problem. On the other hand, though SVI makes significant efficiency improvement over existing solutions, it still takes days to process a network with 58K vertices, which is too slow to be used in practice for large networks.

Besides VB, there has been the development of other approaches to SBM inference including MCMC [2], [23] and belief propagation [24]. However, they are either considered computationally demanding in general or suffer from convergence issues [25]. None of them has demonstrated the success in handling large networks.

Recently, the natural conjugate gradient method has attracted rising attention and has been successfully used to accelerate VB inference for various probabilistic models, such as the nonlinear state-space model for temporal data [26], Gaussian mixture model for vectorial data [27], and latent Dirichlet allocation for textual data [28]. To the best of our knowledge, we are the first to introduce this optimization technique to network data.

VII. Conclusions

We developed a fast inference algorithm called NCG-VB for stochastic blockmodel. Our algorithm utilizes the natural conjugate gradient method to enable the efficient batch optimization of local variational parameters, making it possible to scale up SBM to fast-growing networks nowadays. The experimental results show that, when used for community detection, our algorithm runs up to two orders of magnitude

[8]Note that the direction of the y axis in Figure 9(a) is reversed.

faster than the state-of-the-art VB algorithm without sacrifying much the solution quality. Compared with the heuristic variant of VB that uses asynchronous update for speedup, our algorithm converges more quickly and always obtains communities with better quality. As the future work, we will extend and investigate the applicability of NCG-VB to other network analysis tasks such as link prediction, attributed graph clustering, dynamic graph modeling, etc.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work is partly supported by the SUG Grant M4081416.020 from Nanyang Technological University.

References

[1] P. W. Holland, K. Laskey, and S. Leinhardt, “Stochastic blockmodels: First steps,” *Social Networks*, vol. 5, no. 2, pp. 109–137, 1983.

[2] K. Nowicki and T. A. Snijders, “Estimation and prediction for stochastic blockstructures,” *Journal of the American Statistical Association*, vol. 96, no. 455, pp. 1077–1087, 2001. [Online]. Available: http://dx.doi.org/10.1198/016214501753208735

[3] J. M. Hofman and C. H. Wiggins, “Bayesian approach to network modularity,” *Physical Review Letters*, vol. 100, no. 25, p. 258701, 2008. [Online]. Available: http://arxiv.org/abs/0709.3512

[4] B. Karrer and M. E. J. Newman, “Stochastic blockmodels and community structure in networks,” *CoRR*, vol. abs/1008.3926, 2010.

[5] A. Clauset, C. Moore, and M. E. Newman, “Hierarchical structure and the prediction of missing links in networks.” *Nature*, vol. 453, no. 7191, pp. 98–101, 2008, automatic medline import.

[6] L. Lu and T. Zhou, “Link prediction in complex networks: A survey,” *Physica A*, vol. 390, no. 6, p. 11501170, 2011.

[7] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng, “A model-based approach to attributed graph clustering,” in *SIGMOD Conference*, 2012, pp. 505–516.

[8] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin, “A bayesian approach toward finding communities and their evolutions in dynamic social networks,” in *SDM*, 2009, pp. 990–1001.

[9] J.-J. Daudin, F. Picard, and S. Robin, “A mixture model for random graphs,” *Statistics and Computing*, vol. 18, no. 2, pp. 173–183, 2008.

[10] P. Latouche, E. Birmelé, and C. Ambroise, “Bayesian methods for graph clustering,” in *GfKl*, 2008, pp. 229–239.

[11] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer, 2006.

[12] A. Honkela, T. Raiko, M. Kuusela, M. Tornio, and J. Karhunen, “Approximate riemannian conjugate gradient learning for fixed-form variational bayes,” *Journal of Machine Learning Research*, vol. 11, pp. 3235–3268, 2010.

[13] S. ichi Amari, “Natural gradient works efficiently in learning,” *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998.

[14] S. Amari and H. Nagaoka, *Methods of Information Geometry*, ser. Translations of Mathematical monographs. Oxford University Press, 2000, vol. 191.

[15] M. J. Beal, “Variational algorithms for approximate bayesian inference,” Ph.D. dissertation, Gatsby Computational Neuroscience Unit, University College London, 2003.

[16] G. Casella and R. Berger, *Statistical inference*. Duxbury Press Belmont, Calif, 1990.

[17] M. J. Wainwright and M. I. Jordan, “Graphical models, exponential families, and variational inference,” *Found. Trends Mach. Learn.*, vol. 1, no. 1-2, pp. 1–305, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1561/2200000001

[18] M. E. J. Newman, “Assortative mixing in networks,” *Phys. Rev. Lett.*, vol. 89, p. 208701, 2002. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.89.208701

[19] P. Gopalan, D. M. Mimno, S. Gerrish, M. J. Freedman, and D. M. Blei, “Scalable inference of overlapping communities,” in *NIPS*, 2012, pp. 2258–2266.

[20] M. Schmidt, “minFunc,” http://www.di.ens.fr/~mschmidt/Software/minFunc.html, 2012, [Online; accessed 20-June-2014].

[21] M. E. J. Newman and M. Girvan, “Finding and evaluating community structure in networks,” *Physical Review E*, vol. 69, p. 066113, 2004.

[22] J. Leskovec, K. J. Lang, and M. W. Mahoney, “Empirical comparison of algorithms for network community detection,” in *WWW*, 2010, pp. 631–640.

[23] A. F. McDaid, T. B. Murphy, N. Friel, and N. J. Hurley, “Improved bayesian inference for the stochastic block model with application to large networks,” *Computational Statistics & Data Analysis*, vol. 60, pp. 12–31, 2013.

[24] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, “Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications,” *CoRR*, vol. abs/1109.3041, 2011.

[25] J. S. Yedidia, W. T. Freeman, and Y. Weiss, “Understanding belief propagation and its generalizations,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

[26] A. Honkela, M. Tornio, T. Raiko, and J. Karhunen, “Natural conjugate gradient in variational inference,” in *ICONIP (2)*, 2007, pp. 305–314.

[27] M. Kuusela, T. Raiko, A. Honkela, and J. Karhunen, “A gradient-based algorithm competitive with variational bayesian em for mixture of gaussians,” in *IJCNN*, 2009, pp. 1688–1695.

[28] J. Hensman, M. Rattray, and N. D. Lawrence, “Fast variational inference in the conjugate exponential family,” in *NIPS*, 2012, pp. 2897–2905.