A Model-based Approach to Attributed Graph Clustering

Zhiqiang Xu
School of Computer Engineering
Nanyang Technological University, Singapore
zxu1@e.ntu.edu.sg

Yiping Ke
Institute of High Performance Computing
Singapore
keyp@ihpc.a-star.edu.sg

Yi Wang
Department of Computer Science
National University of Singapore
wangy@comp.nus.edu.sg

Hong Cheng
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
hcheng@se.cuhk.edu.hk

James Cheng
School of Computer Engineering
Nanyang Technological University, Singapore
j.cheng@acm.org

ABSTRACT

Graph clustering, also known as community detection, is a long-standing problem in data mining. However, with the proliferation of rich attribute information available for objects in real-world graphs, how to leverage structural and attribute information for *clustering attributed graphs* becomes a new challenge. Most existing works take a distance-based approach. They proposed various distance measures to combine structural and attribute information. In this paper, we consider an alternative view and propose a model-based approach to attributed graph clustering. We develop a Bayesian probabilistic model for attributed graphs. The model provides a principled and natural framework for capturing both structural and attribute aspects of a graph, while avoiding the artificial design of a distance measure. Clustering with the proposed model can be transformed into a probabilistic inference problem, for which we devise an efficient variational algorithm. Experimental results on large real-world datasets demonstrate that our method significantly outperforms the state-of-art distance-based attributed graph clustering method.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining; G.2.2 [**Graph Theory**]: Graph Algorithms

General Terms

Algorithms, Experimentation, Performance

Keywords

Attributed graph clustering, model-based clustering, Bayesian method

SIGMOD'12, May 20–24, 2012, Scottsdale, Arizona, USA.

1. INTRODUCTION

Graph clustering is a well-studied problem in data mining and machine learning. The objective of graph clustering is to group vertices in a given graph based on vertex connections (i.e., edges). In recent years, with the proliferation of rich information available for real-world objects, vertices in graphs are often associated with a number of attributes that describe the characteristics and properties of the vertices. This gives rise to a new type of graphs, namely *attributed graphs*, and hence the demand of a new clustering task, *attributed graph clustering*.

Attributed graph clustering is useful in many application domains. For example, in service-oriented social networking sites (e.g., Facebook, Twitter, LinkedIn), users and their interactions (e.g., friend-of, follower-of) form a social network. Each user in the network can be further characterized by various information in his/her personal profile, such as interests, gender, education, residency, etc. Clustering the users in a social network by considering both their global social relationships and personal profiles is particularly useful for social networking sites in service/apps recommendation, user-targeted online advertising, etc. In telecommunication business, a communication network consists of subscribers as vertices and their communications (e.g., voice calls, text messaging, email) as edges. Each subscriber is associated with attributes such as demographic information, current service plans, service usage, etc. User groups discovered by clustering the attributed communication network can be used to design effective group-oriented marketing strategies so as to mitigate customer churns for telecom operators.

Attributed graph clustering, though having many important applications, poses significant new challenges. An attributed graph contains two completely different types of information, structural connections and attribute values. Traditional graph clustering and object clustering handle only one of the two types, and thus the resultant clustering inevitably does not reflect both information in the input graph. Therefore, in the problem of clustering an attributed graph, how to naturally combine and leverage these two types of information in the process of clustering becomes a big challenge.

Existing works [19, 1, 20, 9, 16] on attributed graph clustering are mainly *distance-based*, which requires to deliberately design a distance measure that can take both structural and attribute information into consideration. Such a distance measure assigns, either

explicitly or implicitly, weights to structural and attribute information in order to achieve a good balance between them. Manual weighting obviously cannot work for a general graph, while learning weights for specific graphs is often time consuming.

In this paper, we consider an alternative view and propose a *model-based* approach to attributed graph clustering. Our approach avoids the artificial design of a distance measure. Instead, it is based on a probabilistic model which fuses both structural and attribute information in a *natural and principled* manner.

Our model is grounded on two assumptions that are well recognized in clustering research: (1) there exists a *true but unknown* clustering of the vertices underlying the data; (2) vertices from the same cluster behave similarly to each other, while vertices from different clusters can behave differently. In our model, the cluster label of each vertex is explicitly represented as a *hidden* variable. Moreover, the model enforces the intra-cluster similarity by asserting that the attribute values and edge connections of a vertex should depend on its cluster label. In particular, for vertices from the same cluster, their attribute values and edge connections should follow the common distributions that are specific to that cluster. Via the hidden clustering variable, we seamlessly leverage the attribute and connection information of the vertices.

Our probabilistic model essentially defines a joint probability distribution over the space of all possible clusterings and all possible attributed graphs. For a given attributed graph to be clustered, the model assigns a probability for each possible clustering of the vertices. Therefore, the clustering problem can be transformed into a standard *probabilistic inference* problem, i.e., to find the clustering that gives the highest probability [12]. Intuitively, this clustering best explains the observed attribute values and edge connections of the graph.

Despite the conceptual simplicity of the inference problem, it is computationally intractable. To address this issue, we take a variational approach [7] and propose an efficient approximate algorithm for probabilistic inference. The idea is to (1) restrict ourselves to a computationally tractable family of distributions, (2) find the distribution from this family that best approximates our probabilistic model, and (3) perform clustering based on this approximation. We formally define the tractable family and formulate the approximation problem as an optimization problem. We further derive the stationary point conditions for the optimization problem, based on which we then develop an iterative optimization procedure. The procedure essentially starts from an initial guess of the clustering, and iteratively improves the quality of the clustering until convergence.

We summarize the main contributions of this paper as follows.

- We propose a novel model-based approach for the problem of attributed graph clustering. Our model conforms to the two fundamental assumptions in clustering research and provides a natural and principled way of leveraging the structural and attribute information in clustering.
- We formulate the clustering problem as a probabilistic inference problem and analyze its computational difficulty. We then take a variational approach and design an efficient approximate algorithm to solve the inference problem.
- We evaluate the performance of our algorithm on real-world attributed graphs. Compared with the state-of-the-art distance-based attributed graph clustering algorithm [20], our algorithm obtains significantly higher clustering quality, in terms of both structural and attribute information, for all datasets tested. Regarding clustering efficiency, our algorithm is consistently faster, from a few times in a small dataset to two orders of magnitude in larger datasets, and consumes substantially less memory.

Organizations. The rest of the paper is organized as follows. Section 2 defines the problem of attributed graph clustering. Section 3 presents our Bayesian model for attributed graph clustering. Section 4 presents our variational inference algorithm. Section 5 reports the experimental results. Section 6 discusses related work. Finally, Section 7 concludes the paper.

2. PROBLEM STATEMENT

An attributed graph G is defined as a 4-tuple (V, E, Λ, F), where $V = \{v_1, v_2, \ldots, v_N\}$ is a set of N vertices, $E = \{(v_i, v_j) : 1 \leq i, j \leq N,\ i \neq j\}$ is a set of edges, $\Lambda = \{a^1, a^2, \ldots, a^T\}$ is a set of T categorical attributes, $F = \{f_1, f_2, \ldots, f_T\}$ is a set of T functions and each $f_t : V \rightarrow dom(a^t)$ assigns each vertex in V an attribute value in the domain $dom(a^t)$ of the attribute a^t (for $1 \leq t \leq T$). In an attributed graph G, a vertex $v_i \in V$ is essentially associated with an attribute vector of length T, where the t-th element in the vector is given by the function $f_t(v_i)$.

In this paper, we restrict our discussions on *undirected* attributed graphs, while our method can be easily extended to process directed attributed graphs.

Given an attributed graph G and the number of clusters K, the clustering problem studied in this paper is to partition the vertex set V of G into K disjoint subsets $V_1, V_2, \ldots, V_K$, where $V = \bigcup_{i=1}^{K} V_i$ and $V_i \cap V_j = \emptyset$ for any $i \neq j$, such that: (1) in terms of structure (i.e., edge connections in G), the vertices within clusters are densely connected, while the vertices in different clusters are sparsely connected; and (2) in terms of attribute (i.e., attribute values on the vertices in G), the vertices within clusters have low diversity in their attribute values, while the vertices in different clusters may have diverse attribute values.

3. A BAYESIAN MODEL FOR ATTRIBUTED GRAPH CLUSTERING

In this section, we present a Bayesian probabilistic model for attributed graph clustering. Given a set of vertices V, a set of attributes Λ, and the number of clusters K, the model defines a joint probability distribution over the space of all possible attributed graphs *and* all possible partitions over V. That is, it defines a probability for each possible combination of attributed graphs and vertex clusterings of the graphs.

Based on this model, we can cluster a given attributed graph by probabilistic inference. Specifically, we calculate the posterior distribution over all possible clusterings of the given graph, and find the most probable clustering with the maximum probability. Intuitively, this clustering best explains the attribute values and edge patterns of the given graph.

We start by introducing some basic notions and notations in Section 3.1. We then present an intuitive generative process for attributed graphs in Section 3.2. This process essentially draws samples of attributed graphs from an underlying distribution. In Section 3.3, we investigate a number of assumptions made by the generative process, based on which we formally define our model. Finally, in Section 3.4, we elaborate how to use this model for attributed graph clustering and analyze the computational challenges.

3.1 Notions and Notations

Suppose we are given a set of vertices V, a set of attributes Λ, and the number of clusters K. Let N and T be the sizes of V and Λ, respectively.

Procedure 1 Generate Clustered Attributed Graph

Input: a set of vertices V, a set of attributes Λ, and the number of clusters K

Output: a sample of clustered attributed graph $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$

1. Choose $\alpha \sim \texttt{Dirichlet}(\xi)$
2. For each cluster $k \in \{1, 2, \ldots, K\}$:
   (a) For each attribute a^t:
   • Choose $\theta_k^t \sim \texttt{Dirichlet}(\gamma^t)$
   (b) For each cluster $l \in \{k, k+1, \ldots, K\}$:
   • Choose $\phi_{kl} \sim \texttt{Beta}(\mu, \nu)$
3. For each vertex $v_i \in V$:
   (a) Choose $Z_i \sim \texttt{Multinomial}(\alpha)$
   (b) For each attribute a^t:
   • Choose $Y_i^t \sim \texttt{Multinomial}(\theta_{Z_i}^t)$
   (c) For each vertex $v_j \in V$ with $i > j$:
   • Choose $X_{ij} \sim \texttt{Bernoulli}(\phi_{Z_i Z_j})$

- An *adjacency matrix* $\mathbf{X} = [X_{ij}]$ is an $N \times N$ symmetric random matrix. Each element X_{ij} is a binary random variable that takes value 0 or 1, which indicates whether there is an edge between vertices v_i and v_j.
- An *attribute matrix* $\mathbf{Y} = [Y_i^t]$ is an $N \times T$ random matrix. Each element Y_i^t is a categorical random variable that takes value from $dom(a^t)$, which denotes the value of attribute a^t associated with vertex v_i.
- A *clustering of vertices* $\mathbf{Z} = [Z_i]$ is an $N \times 1$ random vector. Each element Z_i is a categorical random variable that takes value from $\{1, 2, \ldots, K\}$, which denotes the label of the cluster that vertex v_i belongs to.

By enumerating the values of $\mathbf{X}$ and $\mathbf{Y}$, we can exhaust all possible attributed graphs over V. Every instantiation of $\mathbf{X}$ and $\mathbf{Y}$ leads to a unique graph. Therefore, we can equivalently represent an attributed graph as a pair $(\mathbf{X}, \mathbf{Y})$. Suppose we further know the value of $\mathbf{Z}$. In this case, we have a clustering (or partition) of the vertex set V. Therefore, we refer to the tuple $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ as a *clustered attributed graph*.

3.2 A Generative Process

Procedure 1 outlines a generative process for clustered attributed graphs. The process takes as input a set of vertices V, a set of attributes Λ, and the number of clusters K. It outputs a sample from all possible clustered attributed graphs, denoted by $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$.

We start by discussing Step 3, the core step of this process. Steps 1 and 2 are for the Bayesian treatment of the model parameters. We will elaborate on them in Section 3.2.2.

3.2.1 Generating $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$

In order to generate a sample of clustered attributed graphs, we need to determine (1) an adjacency matrix $\mathbf{X} = [X_{ij}]$, (2) an attribute matrix $\mathbf{Y} = [Y_i^t]$, and (3) a clustering of vertices $\mathbf{Z} = [Z_i]$. At Step 3 of Procedure 1, we generate them as follows:

1. We first sample the cluster label Z_i of each vertex v_i from a multinomial distribution independently (Step 3(a)). The multinomial distribution is defined as

$$p(Z_i = k|\alpha) = \alpha_k, \quad k = 1, 2, \ldots, K. \tag{1}$$

   The distribution is parameterized by a K-vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_K)$. The element α_k denotes the proportion of the vertices belonging to cluster k, and satisfies the constraints $\alpha_k \in [0, 1]$ and $\sum_{k=1}^K \alpha_k = 1$. For now, let us assume the parameter α (and also θ, ϕ below) is given. Later on, we will show how to sample α from a Bayesian prior distribution.

2. Given the cluster label Z_i for vertex v_i, we then sample the attribute values of this vertex (Step 3(b)). Specifically, we sample the value Y_i^t of each attribute a^t from a multinomial distribution defined as

$$p(Y_i^t = m|\theta_{Z_i}^t) = \theta_{Z_i m}^t, \quad m = 1, 2, \ldots, M^t. \tag{2}$$

   The parameter of the distribution is a M^t-vector $\theta_{Z_i}^t = (\theta_{Z_i 1}^t, \theta_{Z_i 2}^t, \ldots, \theta_{Z_i M^t}^t)$, where M^t is the size of the domain $dom(a^t)$. The element $\theta_{Z_i m}^t$ denotes the proportion of vertices in cluster Z_i that take the m-th value in $dom(a^t)$. It satisfies $\theta_{Z_i m}^t \in [0, 1]$ and $\sum_{m=1}^{M^t} \theta_{Z_i m}^t = 1$.

   As indicated by its subscript Z_i of $\theta_{Z_i}^t$, the multinomial distribution is specific to cluster Z_i. In other words, all vertices belonging to the same cluster share a common multinomial distribution, while the distributions can differ across different clusters. The idea is that vertices in the same cluster are similar to each other. Therefore, they should exhibit a similar pattern in their attribute values.

3. Given the cluster labels Z_i and Z_j for vertex pair v_i and v_j, we finally sample the indicator X_{ij} which denotes whether there is an edge between v_i and v_j (Step 3(c)). X_{ij} is a binary variable taking value 0 or 1. We sample it from a Bernoulli distribution defined as

$$p(X_{ij}|\phi_{Z_i Z_j}) = \left(1 - \phi_{Z_i Z_j}\right)^{1 - X_{ij}} \left(\phi_{Z_i Z_j}\right)^{X_{ij}}. \tag{3}$$

   The parameter $\phi_{Z_i Z_j}$ denotes the edge occurrence probability between clusters Z_i and Z_j, and satisfies $\phi_{Z_i Z_j} \in [0, 1]$ and $\phi_{Z_i Z_j} = \phi_{Z_j Z_i}$.

   Note that the parameter $\phi_{Z_i Z_j}$ depends on the cluster labels Z_i and Z_j. The implication is as follows. Consider two vertices v_i and v_j. Suppose we are generating the indicators X_{ik} and X_{jk} for these vertices with respect to a common third vertex v_k. If v_i and v_j come from the same cluster, i.e., $Z_i = Z_j$, we sample X_{ik} and X_{jk} from the same Bernoulli distribution. Otherwise, we use different Bernoulli distributions for sampling. This is reasonable because vertices from the same cluster should be similar to each other, and they should have the same chance to connect with other vertices. On the other hand, for vertices from different clusters, the chance may diverge.

3.2.2 Generating α, θ, ϕ

So far we assume that the parameters α, θ, and ϕ are given in the generative process. However, the behavior of the process depends on the choice of these parameters. We now discuss how to specify the parameter values.

We take a Bayesian approach to address this issue. Instead of presuming a fixed value for each parameter, we treat α, θ, and ϕ themselves as random variables and place a *prior* distribution over them. By doing so, we explicitly model the intrinsic uncertainty in the values of α, θ, and ϕ. We essentially take all possible values into consideration rather than sticking to a single hypothetic value. We will discuss more about the benefits of the Bayesian treatment in Section 3.4.

To obtain the values of α, θ, and ϕ required by Step 3 of Procedure 1, we sample those values from the prior distribution (Steps 1 and 2 of Procedure 1) as follows.

1. We place a Dirichlet distribution over α, from which we sample the value of α at Step 1. The density function of the Dirichlet distribution is defined as

$$p(\alpha|\xi) = \frac{\Gamma\left(\sum_{k=1}^{K} \xi_k\right)}{\prod_{k=1}^{K} \Gamma\left(\xi_k\right)} \prod_{k=1}^{K} \alpha_k^{\xi_k - 1}, \quad (4)$$

   where $\Gamma(\cdot)$ is the Gamma function. The distribution is parameterized by a positive real K-vector $\xi = (\xi_1, \xi_2, \ldots, \xi_K)$. We refer to ξ as a *hyper-parameter* of the generative process in order to distinguish it from the parameter α.

   The choice of the Dirichlet distribution for α (and also the distributions for θ, ϕ below) is not arbitrary. We will elaborate on this point at the end of this subsection.

2. We place a Dirichlet distribution over θ_k^t for each attribute a^t and each cluster k, defined as

$$p(\theta_k^t|\gamma^t) = \frac{\Gamma\left(\sum_{m=1}^{M^t} \gamma_m^t\right)}{\prod_{m=1}^{M^t} \Gamma\left(\gamma_m^t\right)} \prod_{m=1}^{M^t} \left(\theta_{km}^t\right)^{\gamma_m^t - 1}. \quad (5)$$

   Similar to ξ, $\gamma^t = (\gamma_1^t, \gamma_2^t, \ldots, \gamma_{M^t}^t)$ is a positive real M^t-vector and is a hyper-parameter of the generative process. Note that there is one dedicated hyper-parameter γ^t for each attribute a^t. This is because the domains of different attributes are different.

   On the other hand, one may notice that γ^t does not depend on the cluster label k, which means that a common set of hyper-parameters are shared by all the K clusters. This should not be interpreted as that the parameters θ_k^t for different clusters are tied to the same value, and thus vertices in different clusters are confined to the same pattern of attribute values. In fact, at Step 2(a) of Procedure 1, we sample the parameter θ_k^t for each cluster k independently. Therefore, the values of θ_k^t vary across different clusters. This allows us to accommodate the inter-cluster heterogeneity of attribute values.

3. Finally, we place a Beta distribution over ϕ_{kl}, from which we sample the value of ϕ_{kl} for each cluster pair $(k, l), k \leq l$ independently (Step 2(b)). The Beta distribution is a special case of Dirichlet distribution with only two components, and is defined as

$$p(\phi_{kl}|\mu, \nu) = \frac{\Gamma(\mu + \nu)}{\Gamma(\mu)\Gamma(\nu)} \phi_{kl}^{\mu - 1} \left(1 - \phi_{kl}\right)^{\nu - 1}. \quad (6)$$

   Here μ and ν are the hyper-parameters of the generative process.

As aforementioned, the choices of Dirichlet and Beta as the prior distributions of α, θ, ϕ are not arbitrary. Recall that we sample $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ from multinomial and Bernoulli distributions parameterized by α, θ, ϕ. Mathematically, the Dirichlet and Beta distributions are known as the *conjugate priors* for multinomial and Bernoulli distributions, respectively [3]. It means that, if the prior distributions of α, θ, ϕ are Dirichlet and Beta, then after we observe an instantiation of $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ (or equivalently, a clustered attributed graph) the posterior distributions of α, θ, ϕ are still Dirichlet and Beta. This will give rise to a closed-form expression for the posterior and thus mathematical convenience when we derive an attributed graph clustering algorithm later.

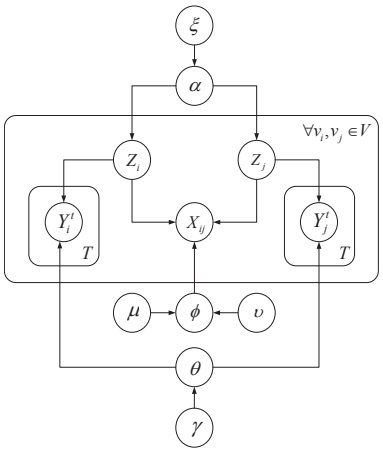


Figure 1: A graphical representation of the proposed model.

In the generative process, all the hyper-parameters are fixed at a predefined value. In this paper, we follow the convention in Bayesian statistics and set the value at 1, which leads to the well known *non-informative* priors [3]. In this case, the Dirichlet and Beta priors over α, θ, ϕ are equivalent to uniform distributions. They assign equal probabilities to all possible values of α, θ, ϕ. Intuitively, it reflects that our prior belief has no preference on any parameter value over the others.

3.3 Model Definition

In the previous subsection, we described a generative process for clustered attributed graphs. Each run of this process generates a sample of the parameters α, θ, ϕ and a sample of clustered attributed graphs $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$. It essentially draws samples from an underlying joint probability distribution over $\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$. In this subsection, we formally define a Bayesian model that represents this underlying distribution.

We first note that the generative process implicitly makes a number of conditional independence assumptions among $\xi, \gamma, \mu, \nu, \alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$. Mathematically, two random variables A and B are conditionally independent given a third variable C if and only if

$$p(A, B|C) = p(A|C)p(B|C).$$

Instead of enumerating the assumptions, we give a compact graphical representation in Figure 1. Each node in the graph corresponds to one random variable. Each rectangle denotes the repetition of the enclosed structure, where the number of repetitions is indicated by the subscript/superscript of the rectangle. For example, the rectangle encompassing the node Y_i^t means that there are T nodes $Y_i^1, Y_i^2, \ldots, Y_i^T$, each of which has 2 parent nodes θ and Z_i. Note that the structure of the graph is constructed according to the flow of the generative process.

The set of conditional independence assumptions can be readily read off the graph. Specifically, a node is independent of all its non-descendants given its parent nodes [12]. For example, Z_i is independent of $\xi, \gamma, \mu, \nu, \theta, \phi$ given α. This is because the generation of Z_i depends only on α (see Step 3(a) of Procedure 1). Similarly, Z_i and Z_j are conditionally independent given α. This is because the cluster labels for different vertices v_i and v_j are sampled independently.

Given the hyper-parameters ξ, γ, μ, ν, we decompose the joint distribution over $\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$ using the probability chain rule and apply the conditional independence assumptions encoded in

Figure 1. This leads to our Bayesian probabilistic model for clustered attributed graphs:

$$\begin{aligned} & p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z} | \xi, \gamma, \mu, \nu) \\ = \; & p(\alpha|\xi) p(\theta|\gamma) p(\phi|\mu, \nu) p(\mathbf{Z}|\alpha) p(\mathbf{X}|\mathbf{Z}, \phi) p(\mathbf{Y}|\mathbf{Z}, \theta), \end{aligned}$$

where

$$\begin{aligned} p(\theta|\gamma) &= \prod_{k=1}^{K} \prod_{t=1}^{T} p(\theta_k^t | \gamma^t), \\ p(\phi|\mu, \nu) &= \prod_{\substack{k,l=1 \\ k \le l}}^{K} p(\phi_{kl}|\mu, \nu), \\ p(\mathbf{Z}|\alpha) &= \prod_{i=1}^{N} p(Z_i|\alpha), \\ p(\mathbf{X}|\mathbf{Z}, \phi) &= \prod_{\substack{i,j=1 \\ i<j}}^{N} p(X_{ij}|\phi_{Z_i Z_j}), \\ p(\mathbf{Y}|\mathbf{Z}, \theta) &= \prod_{i=1}^{N} \prod_{t=1}^{T} p(Y_i^t | \theta_{Z_i}^t), \end{aligned}$$

and $p(Z_i|\alpha)$, $p(Y_i^t|\theta_{Z_i}^t)$, $p(X_{ij}|\phi_{Z_i Z_j})$, $p(\alpha|\xi)$, $p(\theta_k^t|\gamma^t)$, $p(\phi_{kl}|\mu, \nu)$ are defined in Equations (1)–(6), respectively.

For brevity, we will omit the conditional part of the joint distribution $p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z}|\xi, \gamma, \mu, \nu)$ and abbreviate it to $p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z})$ in the rest of this paper. The same applies to all the conditional and marginal distributions that are derived from this joint distribution. One should however always bear in mind that all these distributions are conditioned on the hyper-parameters ξ, γ, μ, ν.

3.4 Model-based Clustering of Attributed Graph

The Bayesian model proposed in the previous subsection defines a joint distribution $p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z})$. Based on this model, the problem of clustering a given attributed graph $(\mathbf{X}, \mathbf{Y})$ can be transformed into a standard probabilistic inference problem, namely, finding the *maximum a posteriori* (MAP) configuration [12] of the clustering $\mathbf{Z}$ conditioning on $\mathbf{X}, \mathbf{Y}$. That is to find

$$\mathbf{Z}^\star = \arg\max_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \mathbf{Y}), \tag{7}$$

where $p(\mathbf{Z}|\mathbf{X}, \mathbf{Y})$ is the posterior distribution of $\mathbf{Z}$ given $\mathbf{X}, \mathbf{Y}$ (and ξ, γ, μ, ν). Intuitively, $\mathbf{Z}^\star$ gives the most probable clustering of the vertex set V that best explains the attribute values $\mathbf{Y}$ and edge patterns $\mathbf{X}$ of the given graph.

Despite its conceptual simplicity, the probabilistic inference problem is notoriously hard. There are two major difficulties.

The first difficulty is the maximization over the N variables $\mathbf{Z} = \{Z_1, Z_2, \ldots, Z_N\}$. For large N, the global maximization is computationally prohibitive.

The second difficulty lies in the calculation of the posterior distribution of $\mathbf{Z}$,

$$p(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) = \iiint p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y}) \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi, \tag{8}$$

where

$$p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y}) = \frac{p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z})}{\sum_{\mathbf{Z}} \iiint p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z}) \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi}. \tag{9}$$

Due to the integrals over the parameters α, θ, ϕ, there is no closed-form expression for $p(\mathbf{Z}|\mathbf{X}, \mathbf{Y})$.

In the next section, we develop an efficient approximate algorithm to address both problems.

We have described how to perform clustering based on the proposed Bayesian model. Before closing this section, we would like to emphasize the significance of the Bayesian treatment to the clustering method. This can be clearly seen from how we calculate the posterior $p(\mathbf{Z}|\mathbf{X}, \mathbf{Y})$ in Equation (8). By treating the parameters α, θ, ϕ as random variables, we model the intrinsic uncertainty in their values. We essentially consider all possible values of α, θ, ϕ, and take average over them by integration. In this way, the estimation of $p(\mathbf{Z}|\mathbf{X}, \mathbf{Y})$ is more reliable, which thus leads to more robust clustering results.

4. A VARIATIONAL ALGORITHM

4.1 The Basic Idea

We develop an efficient variational algorithm to solve the probabilistic inference problem. The basic idea is to approximate the distribution $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$ defined in Equation (9) using a *variational distribution* $q(\alpha, \theta, \phi, \mathbf{Z})$ that is tractable for the maximization over $\mathbf{Z}$ and integration over α, θ, ϕ in Equations (7) and (8).

Specifically, we restrict the variational distribution to a family of distributions that factorize as follows:

$$q(\alpha, \theta, \phi, \mathbf{Z}) = q(\alpha) q(\theta) q(\phi) \prod_i q(Z_i). \tag{10}$$

We then find the distribution within this family that is the most similar to the truth $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$ as the approximation. Given this approximation $q(\alpha, \theta, \phi, \mathbf{Z})$, we can approximate the MAP clustering $\mathbf{Z}^\star$ as follows:

$$\begin{aligned} \mathbf{Z}^\star &= \arg\max_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) \\ &= \arg\max_{\mathbf{Z}} \iiint p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y}) \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi \\ &\approx \arg\max_{\mathbf{Z}} \iiint q(\alpha, \theta, \phi, \mathbf{Z}) \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi \\ &= \arg\max_{\mathbf{Z}} \iiint q(\alpha) q(\theta) q(\phi) \prod_i q(Z_i) \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi \\ &= \arg\max_{\mathbf{Z}} \prod_i q(Z_i) \\ &= \left[\arg\max_{Z_1} q(Z_1), \arg\max_{Z_2} q(Z_2), \ldots, \arg\max_{Z_N} q(Z_N) \right]. \end{aligned} \tag{11}$$

Due to the factorization of $q(\alpha, \theta, \phi, \mathbf{Z})$, the integrals over α, θ, ϕ diminish and the global maximization over $\mathbf{Z}$ reduces to local maximizations over each Z_i independently.

Two questions remain: (1) how to define the family of variational distributions, and (2) how to find the best distribution from this family that is the closest to $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$. We address these two questions in Sections 4.2 and 4.3, respectively.

4.2 Parametric Family of Variational Distributions

We have put a general restriction on the family of variational distributions in Equation (10). We further require the distributions in this family to take the following parametric form:

$$q(\alpha, \theta, \phi, \mathbf{Z}|\tilde{\xi}, \tilde{\gamma}, \tilde{\mu}, \tilde{\nu}, \tilde{\beta}) = q(\alpha|\tilde{\xi}) q(\theta|\tilde{\gamma}) q(\phi|\tilde{\mu}, \tilde{\nu}) \prod_i q(Z_i|\tilde{\beta}_i),$$

where

$$q(\alpha|\tilde{\xi}) = \mathrm{Dirichlet}(\tilde{\xi}),$$

$$
\begin{aligned}
q(\phi|\tilde{\mu},\tilde{\nu}) &= \prod_{\substack{k,l=1\\k\le l}}^{K} q(\phi_{kl}|\tilde{\mu}_{kl},\tilde{\nu}_{kl}),\\
q(\phi_{kl}|\tilde{\mu}_{kl},\tilde{\nu}_{kl}) &= \mathrm{Beta}(\tilde{\mu}_{kl},\tilde{\nu}_{kl}),\\
q(\theta|\tilde{\gamma}) &= \prod_{k=1}^{K}\prod_{t=1}^{T} q(\theta_k^t|\tilde{\gamma}_k^t),\\
q(\theta_k^t|\tilde{\gamma}_k^t) &= \mathrm{Dirichlet}(\tilde{\gamma}_k^t),\\
q(Z_i|\tilde{\beta}_i) &= \mathrm{Multinomial}(\tilde{\beta}_i).
\end{aligned}
$$

Here $\tilde{\xi},\tilde{\gamma},\tilde{\mu},\tilde{\nu},\tilde{\beta}$ are the *variational parameters*. The distributions in this family can be exhausted by enumerating the instantiations of these parameters. Each instantiation specifies a unique distribution.

This definition of the family of variational distributions is not arbitrary. In fact, the distributions $q(\alpha|\tilde{\xi})$, $q(\phi|\tilde{\mu},\tilde{\nu})$, $q(\theta|\tilde{\gamma})$, and $q(Z_i|\tilde{\beta}_i)$ take exactly the same parametric forms as $p(\alpha|\xi)$, $p(\phi|\mu,\nu)$, $p(\theta|\gamma)$, and $p(Z_i|\alpha)$ in our Bayesian model (see Section 3.3). There are only two differences. The first is that the variational parameters $\tilde{\xi},\tilde{\gamma},\tilde{\mu},\tilde{\nu},\tilde{\beta}$ are free to vary, while the hyper-parameters ξ,γ,μ,ν are fixed throughout the clustering process.

The second difference is between $q(Z_i|\tilde{\beta}_i)$ and $p(Z_i|\alpha)$. We introduce a variational parameter $\tilde{\beta}_i$ to replace the parameter α. This is because we require the variational distribution to factorize according to Equation (10). Therefore, Z_i should no longer depend on α with respect to the distribution.

For simplicity, we will omit the conditional parts of the $q(\cdot)$ distributions. For example, we will abbreviate $q(\alpha,\theta,\phi,\mathbf{Z}|\tilde{\xi},\tilde{\gamma},\tilde{\mu},\tilde{\nu},\tilde{\beta})$ and $q(\mathbf{Z}|\tilde{\beta})$ to $q(\alpha,\theta,\phi,\mathbf{Z})$ and $q(\mathbf{Z})$, respectively. One should however always bear in mind that the $q(\cdot)$ distributions are conditioned on the variational parameters $\tilde{\xi},\tilde{\gamma},\tilde{\mu},\tilde{\nu},\tilde{\beta}$.

Before closing this subsection, we would like to point out a consequence of the definition $q(\mathbf{Z})$. Because $q(Z_i)$ is multinomial distribution, the MAP clustering $\mathbf{Z}^\star$ in Equation (11) can be further simplified as follows:

$$
\begin{aligned}
\mathbf{Z}^\star &= \left[\arg\max_{Z_1} q(Z_1), \arg\max_{Z_2} q(Z_2), \ldots, \arg\max_{Z_N} q(Z_N)\right]\\
&= \left[\arg\max_{k} \tilde{\beta}_{1k}, \arg\max_{k} \tilde{\beta}_{2k}, \ldots, \arg\max_{k} \tilde{\beta}_{Nk}\right]. \quad (12)
\end{aligned}
$$

4.3 Optimizing Variational Parameters

Recall that our goal is to find the variational distribution in the family that is the closest to the true posterior $p(\alpha,\theta,\phi,\mathbf{Z}|\mathbf{X},\mathbf{Y})$. This is now equivalent to optimize the variational parameters $\tilde{\xi},\tilde{\gamma},\tilde{\mu},\tilde{\nu},\tilde{\beta}$ with respect to some distance measure. In the following, we define a distance measure and present an iterative optimization procedure.

4.3.1 The Objective Function

To measure the distance between a variational distribution $q(\alpha,\theta,\phi,\mathbf{Z})$ and the true posterior $p(\alpha,\theta,\phi,\mathbf{Z}|\mathbf{X},\mathbf{Y})$, we adopt the Kullback-Leibler (KL) divergence [2] that is commonly used in information theory and machine learning. It is defined as

$$
\mathrm{KL}(q||p) = \sum_{\mathbf{Z}} \iiint q(\alpha,\theta,\phi,\mathbf{Z}) \log \frac{q(\alpha,\theta,\phi,\mathbf{Z})}{p(\alpha,\theta,\phi,\mathbf{Z}|\mathbf{X},\mathbf{Y})} \mathrm{d}\alpha\mathrm{d}\theta\mathrm{d}\phi. \quad (13)
$$

Note that the KL divergence is a function of the variational parameters $\tilde{\xi},\tilde{\gamma},\tilde{\mu},\tilde{\nu},\tilde{\beta}$. Our problem is thus to find the optimal variational parameters that minimize the KL divergence. However, this optimization problem is infeasible because the KL divergence involves the term $p(\alpha,\theta,\phi,\mathbf{Z}|\mathbf{X},\mathbf{Y})$, which is exactly what we strive to approximate in the first place.

Instead of directly minimizing the KL divergence, we solve an equivalent *maximization* problem. The objective function of this maximization problem is defined as

$$
\widetilde{L}(q) = \sum_{\mathbf{Z}} \iiint q(\alpha,\theta,\phi,\mathbf{Z}) \log \frac{p(\alpha,\theta,\phi,\mathbf{X},\mathbf{Y},\mathbf{Z})}{q(\alpha,\theta,\phi,\mathbf{Z})} \mathrm{d}\alpha\mathrm{d}\theta\mathrm{d}\phi. \quad (14)
$$

The equivalence between these two optimization problems can be easily seen by noticing that their objective functions sum up to a constant:

$$
KL(q||p) + \widetilde{L}(q) = \log p(\mathbf{X},\mathbf{Y}).
$$

4.3.2 Stationary Points of $\widetilde{L}(q)$

In order to maximize the objective function $\widetilde{L}(q)$, we first characterize its stationary points. Specifically, we take the derivatives of $\widetilde{L}(q)$ with respect to the variational parameters $\tilde{\xi},\tilde{\gamma},\tilde{\mu},\tilde{\nu},\tilde{\beta}$, and set these derivatives to zeros

$$
\nabla\widetilde{L}(q) = \left(\frac{\partial\widetilde{L}}{\partial\tilde{\xi}}, \frac{\partial\widetilde{L}}{\partial\tilde{\gamma}}, \frac{\partial\widetilde{L}}{\partial\tilde{\mu}}, \frac{\partial\widetilde{L}}{\partial\tilde{\nu}}, \frac{\partial\widetilde{L}}{\partial\tilde{\beta}}\right) = 0.
$$

Plugging in the definitions of $\widetilde{L}(q)$ and $q(\alpha,\theta,\phi,\mathbf{Z})$ and simplifying the formulas, we arrive at the following system of equations that $\tilde{\xi},\tilde{\gamma},\tilde{\mu},\tilde{\nu},\tilde{\beta}$ must satisfy at the stationary points of $\widetilde{L}(q)$. For clarity, we put all the derivations in Appendix A.

$$
\tilde{\xi}_k = \xi_k + \sum_{i=1}^{N} \tilde{\beta}_{ik} \quad (15)
$$

$$
\tilde{\gamma}_{km}^t = \gamma_m^t + \sum_{i=1}^{N} \tilde{\beta}_{ik}\delta\left(Y_i^t, a_m^t\right) \quad (16)
$$

$$
\tilde{\mu}_{kk} = \mu + \sum_{\substack{i,j=1\\i<j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jk}X_{ij} \quad (17)
$$

$$
\tilde{\nu}_{kk} = \nu + \sum_{\substack{i,j=1\\i<j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jk}(1-X_{ij}) \quad (18)
$$

$$
\tilde{\mu}_{kl} = \mu + \sum_{\substack{i,j=1\\i\neq j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jl}X_{ij} \quad (19)
$$

$$
\tilde{\nu}_{kl} = \nu + \sum_{\substack{i,j=1\\i\neq j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jl}(1-X_{ij}) \quad (20)
$$

$$
\begin{aligned}
\tilde{\beta}_{ik} \propto \exp\Bigg\{ &\left[\psi\left(\tilde{\xi}_k\right) - \psi\left(\sum_{k=1}^{K}\tilde{\xi}_k\right)\right]\\
&+ \sum_{t=1}^{T}\sum_{m=1}^{M^t} \delta(Y_i^t, a_m^t)\left[\psi\left(\tilde{\gamma}_{km}^t\right) - \psi\left(\sum_{m=1}^{M^t}\tilde{\gamma}_{km}^t\right)\right]\\
&+ \sum_{\substack{j=1\\j\neq i}}^{N}\sum_{l=1}^{K} \tilde{\beta}_{jl}\Big[X_{ij}\psi(\tilde{\mu}_{kl}) + (1-X_{ij})\psi(\tilde{\nu}_{kl})\\
&-\psi(\tilde{\mu}_{kl}+\tilde{\nu}_{kl})\Big]\Bigg\}, \quad (21)
\end{aligned}
$$

for all $i = 1,2,\ldots,N$; $k = 1,2,\ldots,K$; $l = k+1,k+2,\ldots,K$;

Algorithm 2 Iterative Optimization of $\widetilde{L}(q)$

Input: an initial value $\tilde{\beta}^{(0)}$, a threshold ϵ, a limit on the number of iterations $n_{\max}$
Output: $\tilde{\xi},\ \tilde{\gamma},\ \tilde{\mu},\ \tilde{\nu},\ \tilde{\beta}$

1. $n \leftarrow 0$
2. **repeat**:
   (a) Given $\tilde{\beta}^{(n)}$, update $\tilde{\xi}^{(n+1)}, \tilde{\gamma}^{(n+1)}, \tilde{\mu}^{(n+1)}, \tilde{\nu}^{(n+1)}$ according to Equations (15)–(20)
   (b) Given $\tilde{\xi}^{(n+1)}, \tilde{\gamma}^{(n+1)}, \tilde{\mu}^{(n+1)}, \tilde{\nu}^{(n+1)}, \tilde{\beta}^{(n)}$, update $\tilde{\beta}^{(n+1)}$ according to Equation (21)
   (c) $n \leftarrow n + 1$
   until $\widetilde{L}\left(q^{(n)}\right) - \widetilde{L}\left(q^{(n-1)}\right) < \epsilon$ or $n > n_{\max}$
3. **return** $\tilde{\xi}^{(n)}, \tilde{\gamma}^{(n)}, \tilde{\mu}^{(n)}, \tilde{\nu}^{(n)}, \tilde{\beta}^{(n)}$

$t = 1, 2, \ldots, T$; and $m = 1, 2, \ldots, M^t$. Here,

$$\delta\left(Y_i^t, a_m^t\right) = \begin{cases} 1, & Y_i^t = a_m^t \\ 0, & Y_i^t \neq a_m^t \end{cases}$$

is the Kronecker delta function. $\psi(\cdot)$ is the Digamma function which is the logarithmic derivative of the Gamma function $\Gamma(\cdot)$,

$$\psi(x) = \frac{\mathrm{d}\log\Gamma(x)}{\mathrm{d}x} = \frac{\Gamma'(x)}{\Gamma(x)}.$$

The Digamma function can be efficiently approximated by series expansion and standard implementations exist in popular mathematical libraries such as Matlab.

4.3.3 Iterative Optimization Procedure

Based on the stationary point equations, we present an iterative procedure for maximizing $\widetilde{L}(q)$ in Algorithm 2. It takes as input an initial value $\tilde{\beta}^{(0)}$, a threshold ϵ, and a limit on the number of iterations $n_{\max}$. It outputs the optimized variational parameters $\tilde{\xi},\ \tilde{\gamma},\ \tilde{\mu},\ \tilde{\nu},\ \tilde{\beta}$. Recall that given $\tilde{\beta}$, we can easily find the clustering $\mathbf{Z}^\star$ of the vertices according to Equation (12).

Intuitively, Algorithm 2 starts with an initial guess of $\tilde{\beta}$, and repeatedly enforces the stationary point conditions. Therefore, it should persistently improve the objective function $\widetilde{L}(q)$. This intuition is formalized by the following proposition.

PROPOSITION 1. *For all* $n = 0, 1, \ldots,$

$$\widetilde{L}\left(q^{(n)}\right) \leq \widetilde{L}\left(q^{(n+1)}\right),$$

where $q^{(n)}$ *denotes* $q(\alpha, \theta, \phi, \mathbf{Z}|\tilde{\xi}^{(n)}, \tilde{\gamma}^{(n)}, \tilde{\mu}^{(n)}, \tilde{\nu}^{(n)}, \tilde{\beta}^{(n)})$.

For clarity, we defer the proof to Appendix B.

Because the value of $\widetilde{L}(q)$ is finite, an immediate corollary of Proposition 1 is that the iterative maximization procedure is guaranteed to converge with a finite number of iterations. In particular, it will converge to a local maximum of $\widetilde{L}(q)$. Because $\widetilde{L}(q)$ is non-concave, it can have multiple local maxima. The quality of the local maximum that the iterative process converges to depends on the choice of the initial value $\tilde{\beta}^{(0)}$. We will discuss the initialization issue in the experiments.

At each iteration of Algorithm 2, we alternate between two steps. At Step 2(a), we update $\tilde{\xi}, \tilde{\gamma}, \tilde{\mu}, \tilde{\nu}$ based on $\tilde{\beta}$. At Step 2(b), we update $\tilde{\beta}$ based on $\tilde{\xi}, \tilde{\gamma}, \tilde{\mu}, \tilde{\nu}$. To speed up the convergence of the algorithm, in practice, we choose to perform asynchronous update and execute Step 2(b) multiple times in each iteration. The key observation is that the stationary point equations of $\tilde{\xi}, \tilde{\gamma}, \tilde{\mu}, \tilde{\nu}$ depend only on $\tilde{\beta}$. Therefore, updating $\tilde{\beta}$ more frequently should bring about more improvement in $\widetilde{L}(q)$ and thus fasten the convergence.

5. EXPERIMENTAL STUDY

We evaluate the performance of our algorithm, comparing with the state-of-the-art distance-based attributed graph clustering algorithm, *Inc-Cluster* [20]. Both algorithms were implemented in Matlab and tested on machines with Linux OS, Intel Xeon 2.67GHz CPUs, and 12GB and 256GB of RAM.

5.1 Datasets

We use the three real attributed graphs that are used in the evaluation of *Inc-Cluster* [20].

- **Political Blogs**. The dataset has $1,490$ vertices and $19,090$ edges. Each vertex represents a webblog on US politics and each directed edge represents a hyperlink from one webblog to another. Each vertex is associated with an attribute, indicating the political leaning of the webblog, *liberal* or *conservative*. Since we only consider undirected graphs in this work, we ignore the edge directions in this dataset, which results in $16,715$ undirected edges.

- **DBLP10K**. The dataset is a co-author network extracted from the DBLP Bibliography data. Each vertex represents a scholar and each edge represents a co-author relationship between two scholars. The dataset contains $10,000$ scholars who have published in major conferences in four research fields: database, data mining, information retrieval, and artificial intelligence. Each scholar is associated with two attributes, *prolific* and *primary topic*. The attribute "prolific" has three values: "highly prolific" for the scholars with ≥ 20 publications, "prolific" for the scholars with ≥ 10 and < 20 publications, and "low prolific" for the scholars with < 10 publications. The domain of the attribute "primary topic" consists of 100 research topics extracted by a topic model [5] from a collection of paper titles from the scholars. Each scholar is then assigned a primary topic out of the 100 topics.

- **DBLP84K**. This dataset is a larger DBLP co-author network. It contains $84,170$ scholars in 15 research fields. In addition to the four research fields used in *DBLP10K*, eleven fields are further included: machine learning, computer vision, networking, multimedia, computer systems, simulation, theory, architecture, natural language processing, human-computer interaction, and programming language. This dataset also has two vertex attributes, which are defined in a similar way as in *DBLP10K*.

Table 1 summarizes the characteristics of the three datasets, including the number of vertices $|V|$, the number of edges $|E|$, and the domain size of each of the attributes $|dom(a^i)|$. Among the three datasets, the *Political Blogs* dataset is the smallest, while the other two datasets are much larger. The larger DBLP dataset, i.e., DBLP84K, allows us to see some scalability issue of the algorithms (e.g., *Inc-Cluster* uses 60GB of memory for this dataset).

5.2 Experimental Settings

We detail the experimental settings in this subsection, including the algorithm for comparison, the measures we use to assess the quality of the clustering, as well as the initializations of parameters.

Table 1: Datasets

	$\|V\|$	$\|E\|$	$\|dom(a^1)\|$	$\|dom(a^2)\|$
Political Blogs	1,490	16,715	2	NIL
DBLP10K	10,000	27,867	3	100
DBLP84K	84,170	201,334	3	100

5.2.1 Algorithm for Comparison

We compare our model-based clustering algorithm, denoted as **BAGC** (Bayesian Attributed Graph Clustering), to the state-of-the-art distance-based clustering algorithm, denoted as **Inc-Cluster** [20]. In order to design a distance measure that considers both structure and attributes, Inc-Cluster constructs an augmented graph, which introduces an artificial node for each attribute value and links a vertex in the input G to the artificial node if the vertex takes the corresponding attribute value. A unified distance measure is defined as the random walk score computed from the augmented graph. The k-medoids algorithm is then applied to cluster vertices with the defined distance measure.

5.2.2 Clustering Quality Assessment

Since our objective is to cluster attributed graphs, we assess the quality of the clustering in two aspects, *structure* and *attribute*.

We use *modularity* as a quality measure for structure. Modularity [11] is popularly used in graph clustering to measure the strength of division of a graph into vertex clusters (i.e., communities).

To define modularity, we first introduce the following notions. Given a clustering $\{V_1, \cdots, V_K\}$, let E_{kl} $(k \neq l)$ be the set of inter-cluster edges between V_k and V_l, and E_{kk} the set of intra-cluster edges in V_k. Then the fraction of intra-cluster edges in V_k is defined as $f_{kk} = \frac{|E_{kk}|}{|E|}$ and the fraction of inter-cluster edges between V_k and V_l $(k \neq l)$ is defined as $f_{kl} = f_{lk} = \frac{|E_{kl}|}{2|E|}$ (There are $2|E|$ edges in the denominator because an edge is shared by f_{kl} and f_{lk}). By counting both intra-cluster and inter-cluster edges, the fraction of edges incident to cluster V_k is defined as $\alpha_k = \sum_{l=1}^{K} f_{kl}$.

The modularity is then defined as

$$modularity(V_1, \cdots, V_K) = \sum_{k=1}^{K} \left(f_{kk} - \alpha_k^2\right).$$

Intuitively, if edges were distributed at random, the expected fraction of intra-cluster edges in V_k is α_k^2. By subtracting this expected fraction (i.e., α_k^2) from the true fraction of intra-cluster edges in V_k (i.e., f_{kk}), modularity reflects the concentration of vertices within clusters compared with random distribution of edges between all vertices regardless of clusters. The value of modularity falls within the range of $[-1, 1]$. A positive value indicates that the number of intra-cluster edges exceeds the number expected on a random basis. Therefore, a clustering result with high modularity has dense vertex connections within the same cluster and sparse vertex connections across different clusters.

For attributes, we use *entropy* as a quality measure. Entropy is a well accepted measure, which quantifies the uncertainty of a random variable. In the problem of clustering attributed graphs, entropy can be used to measure the degree of inconsistency of the attribute values in each cluster.

Given a clustering $\{V_1, \cdots, V_K\}$, for each attribute a^t, the entropy of a^t in cluster V_k is defined as

$$\textit{entropy}(a^t, V_k) = -\sum_{s=1}^{|dom(a^t)|} p_{ks}^t \log p_{ks}^t,$$

where p_{ks}^t is the fraction of vertices in cluster V_k that take the s-th value in $dom(a^t)$.

We then define the entropy of an attribute a^t with respect to the clustering $\{V_1, \cdots, V_K\}$ as

$$\textit{entropy}(a^t) = \sum_{k=1}^{K} \frac{|V_k|}{|V|} \textit{entropy}(a^t, V_k),$$

which is the average entropy of K clusters weighted by the cluster size $|V_k|$. The value of entropy falls within the range of $[0, \infty)$. A lower entropy indicates a higher degree of consistency in the attribute values associated with the vertices in the same cluster and thus a higher intra-cluster attribute similarity.

5.2.3 Initializations of Parameters

For the fairness of comparison, BAGC uses the same initial clustering as in Inc-Cluster to initialize $\tilde{\beta}$. Since both algorithms adopt the same initialization process, we report the running time and memory consumption of the optimization process only, by excluding the initialization part to reflect the pure performance of the clustering algorithms. We note that other initialization methods such as random initialization and spectral methods can also be applied in our algorithm.

All hyperparameters in BAGC are set to 1 for all experiments. The threshold ϵ for the objective function $\widetilde{L}(q)$ is set to 10^{-10}. The limit on the number of iterations n_{max} is set to 3 for the small dataset Political Blogs, and 5 for larger datasets DBLP10K and DBLP84K.

5.3 Performance Results

We report and discuss the performance results of both BAGC and Inc-Cluster, for each of the three datasets as follows.

5.3.1 Clustering Performance on Political Blogs

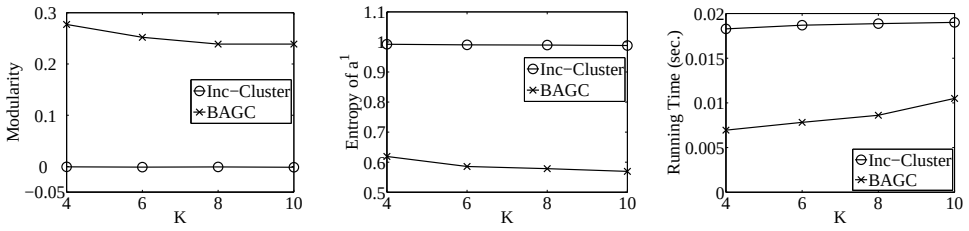
For the Political Blogs dataset, we set the number of clusters, $K = 4, 6, 8,$ and 10, respectively.

We first examine the quality of clustering with respect to structural information. Figure 2(a) reports the modularity of the clustering by BAGC and Inc-Cluster. The result shows that according to the modularity values (the higher the better), BAGC achieves significantly higher quality clustering than Inc-Cluster, since a value of nearly 0.3 is a big difference in modularity [10].

The modularity values of the clusterings by Inc-Cluster are in fact all negative, meaning that the clustering computed by Inc-Cluster has a lower quality than a clustering obtained by randomly distributing edges to different clusters. Note that when the clustering is formed by random chance, the value of modularity is 0. The poor performance of Inc-Cluster is mainly because it is a distance-based method with the objective of optimizing the intra-cluster distance. Their distance measure, i.e., the random walk score, may not be a good reflection of community structures.

Next we assess the quality of clustering with respect to attribute information. Figure 2(b) reports the attribute entropy of the clustering by BAGC and Inc-Cluster on the Political Blogs dataset. On average, BAGC improves the attribute entropy of Inc-Cluster by 40.6%. The much lower entropy value of BAGC shows that BAGC attains a much higher degree of consistency in intra-cluster attribute values, which indicates a much higher attribute similarity than Inc-Cluster.

Our method is able to obtain low attribute entropy because in our generative model, the attribute value of the vertices in the same cluster is drawn from the same multinomial distribution. The process of optimizing $\widetilde{L}(q)$ favors more skewed multinomial distribu-



(a) Modularity (b) Entropy of Attribute 1 (c) Running Time

Figure 2: Clustering Performance on Political Blogs

tion and thus achieves a low attribute entropy in clustering results. On the other hand, Inc-Cluster converts attribute values to artificial nodes and their links to original vertices to form an augmented graph. This increases the vertex connectivity with additional paths through artificial nodes. However, it may not lead to consistent attribute values in the clusters. For example, two vertices u and v are both connected to an artificial node of an attribute value a_1, meaning that both u and v take the attribute value a_1. Then suppose v is structurally connected to another vertex w and w takes a different attribute value a_2. Inc-Cluster may put u, v, w in the same cluster since their random walk scores can be high due to the paths through a_1, even though u, v, w do not exhibit high attribute consistency.

For different values of K tested, the clustering quality of Inc-Cluster is stable. But for BAGC, the modularity becomes slightly smaller (worse) and the entropy also becomes smaller (better) when K increases. This indicates that for different values of K, BAGC is able to leverage the clustering quality in terms of structure and attribute in order to achieve a stable overall performance.

We now report the running time, i.e., the elapsed time in seconds, of clustering Political Blogs by both algorithms, as shown in Figure 2(c). The result shows that BAGC is approximately 2 to 3 times faster than Inc-Cluster on average. Therefore, both efficiency-wise and quality-wise, the results show that BAGC is a clear winner.

5.3.2 Clustering Performance on DBLP10K

For the DBLP10K dataset, we set the number of clusters, $K = 50, 100, 200,$ and 300, respectively.

Figure 3(a) shows that BAGC obtains high quality clustering as the modularity for all values of K is constantly over 0.4. According to Newman [10], a modularity value of 0.3 already indicates a significant community structure, i.e., a high quality clustering. On the contrary, Inc-Cluster records a low modularity for all values of K. The difference in the modularity value between BAGC and Inc-Cluster also becomes greater in this larger dataset than in the smaller Political Blogs dataset.

Figures 3(b) and 3(c) report the entropy values of each of the two attributes[1] of DBLP10K. The results show that BAGC attains considerably lower entropy values for both attributes than Inc-Cluster, demonstrating the advantage of BAGC over Inc-Cluster in attaining a higher quality clustering with respective to attribute information, in addition to structural information.

Figure 3(d) reports the running time of BAGC and Inc-Cluster. The result shows that BAGC is two orders of magnitude faster than Inc-Cluster. As K increases, the running time of BAGC also increases linearly with K (the same trend is also observed in Figures 2(c) and 4(d)). However, for the running time of Inc-Cluster, we observe a different trend. Inc-Cluster uses almost the same amount of time for the range of K from 50 to 100, then there is a sudden increase in running time as K increases to 200, and then the running time remains stable again for the range from 200 to 300. We examined the Inc-Cluster algorithm and found that the number of iterations in its optimization process increases in a step-wise manner as K increases, and the running time is mainly determined by the number of iterations. We also found that Inc-Cluster uses 3 iterations for $K \in [50..100]$ and 5 iterations for $K \in [200..300]$, which explains the trend shown in Figure 3(d). However, as shown in Figure 3(d), the magnitude of increase in the running time of Inc-Cluster is significantly more rapid than the linear increase in that of BAGC.

[1]Note that the entropy value of Attribute 1 is significantly lower than that of Attribute 2 because the domain size of Attribute 1 is significantly smaller than that of Attribute 2, as shown in Table 1.

5.3.3 Clustering Performance on DBLP84K

For the largest dataset, DBLP84K, we use a wider range of values of K, with $K = 150, 300, 600$ and 1200, respectively.

As reported in Figure 4(a), the modularity value of BAGC for this largest dataset is the highest among three datasets, and is consistently over 0.5, implying a very high quality clustering. The difference in the modularity value between BAGC and Inc-Cluster also further widens as the modularity value of Inc-Cluster remains low. The result thus shows that the quality of clustering obtained by BAGC improves over larger datasets.

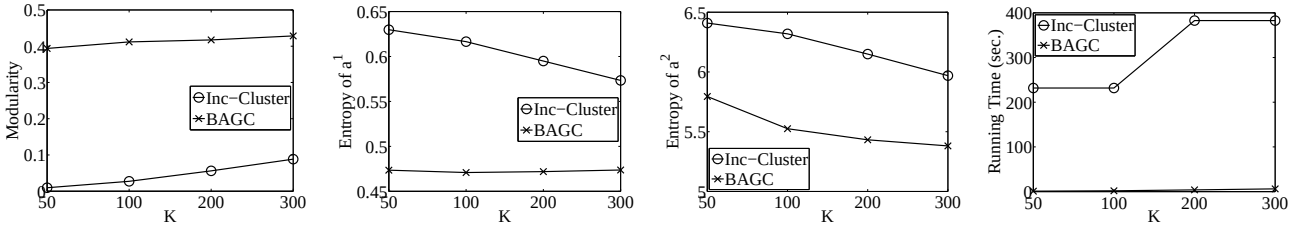
Figures 4(b) and 4(c) further show that BAGC consistently attains lower entropy values for both of the attributes than Inc-Cluster. Thus, the result again demonstrates that BAGC obtains higher quality clustering with more consistent intra-cluster attribute values than Inc-Cluster for large datasets as well.

Finally, Figure 4(d) again shows a huge gap between the running time of BAGC and that of Inc-Cluster. As explained for Figure 3(d) in Section 5.3.2, the running time of BAGC increases linearly as K increases. On the contrary, the running time of Inc-Cluster increases in a step-wise manner, with a significantly greater magnitude in the increase. Thus, the result shows that BAGC is more scalable in clustering large datasets than Inc-Cluster.

5.3.4 Conclusions on Performance Comparison

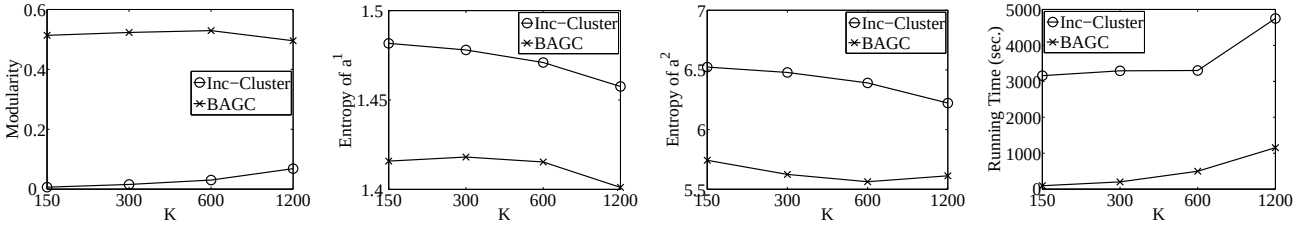
In conclusion, the results in Sections 5.3.1 to 5.3.3 show that BAGC consistently attains high quality clustering in terms of both structure quality and attribute quality. Compared with the state-of-the-art distance-based attributed graph clustering algorithm, Inc-Cluster [20] (an improved version of SA-Cluster [19]), the clustering obtained by BAGC has significantly higher modularity and lower entropy for all datasets and all values of K. In addition, BAGC is also consistently faster than Inc-Cluster, where the speed-up in time is up to two orders of magnitude for the two larger datasets. The results further show that BAGC is much more scalable for clustering larger datasets with larger values of K.

We did not show the details of memory consumption of the algorithms in Sections 5.3.1 to 5.3.3, but report here that Inc-Cluster requires significantly more memory than BAGC. Inc-Cluster uses ap-



(a) Modularity (b) Entropy of Attribute 1 (c) Entropy of Attribute 2 (d) Running Time

Figure 3: Clustering Performance on DBLP10K



(a) Modularity (b) Entropy of Attribute 1 (c) Entropy of Attribute 2 (d) Running Time

Figure 4: Clustering Performance on DBLP84K

proximately 460MB, 4GB, and 60GB of memory, while our method uses 320MB, 680MB, and 4GB of memory for Political Blogs, DBLP10K, and DBLP84K, respectively. The memory consumption, especially a huge difference of 56GB for DBLP84K, also reveals that our method is more scalable than Inc-Cluster when the input dataset becomes large. Note that due to the high memory consumption of Inc-Cluster for DBLP84K, we ran all the experiments of DBLP84K for both Inc-Cluster and BAGC on a computer with 256GB of RAM.

To summarize, with the remarkably higher clustering efficiency (in terms of both time and memory consumption), and the significantly better clustering quality (in terms of both structure and attribute), our model-based approach is evidentially a more promising solution to attributed graph clustering than existing distance-based approaches.

6. RELATED WORK

The algorithms for attributed graph clustering can be mainly categorized into two types, *distance-based* and *model-based*.

Most existing works on attributed graph clustering fall into the category of distance-based approaches. The main idea is to design a distance/similarity measure for vertex pairs that combines both structural and attribute information of the vertices. Based on this measure, standard clustering algorithms like k-medoids and spectral clustering are then applied to cluster the vertices.

Neville et al. [9] proposed a weighted adjacency matrix as the similarity measure. The weight of each edge is defined as the number of attribute values shared by the two end vertices. They then applied three existing graph clustering algorithms on the weighted adjacency matrix to perform clustering. Steinhaeuser and Chawla [16] proposed a similar measure as in [9] to handle categorical attributes, and also a new measure for continuous attributes.

The state-of-the-art distance-based approaches are the SA-Cluster proposed by Zhou et al. [19] and its extended versions, SA-Cluster-Opt [1] and Inc-Cluster [20]. In their work, an augmented graph is constructed by linking all vertices that share the same attribute value to a common artificial node. They then defined the distance measure as the random walk score [17] computed from the augmented graph. In the distance measure, different weights are assigned to structure and attributes, which can be tuned automatically by their algorithm. The K-mediods algorithm is then applied to find the clustering. In order to efficiently compute the distance measure, they further proposed an approximate distance computation in SA-Cluster-Opt [1] and an incremental distance computation in Inc-Cluster [20]. In this paper, we compared our algorithm with Inc-Cluster and as evidenced by the experimental results, our approach significantly outperforms Inc-Cluster in terms of both clustering quality and efficiency.

Although the distance-based approach has been extensively studied, little has been done on the model-based approach to attributed graph clustering. We are only aware of two existing works [18, 4] in this category. [18] adopts a similar generative process as ours, and also proposes a probabilistic model to cluster attributed graphs. However, there are two major differences between [18] and our work. First, their work targets on continuous attributes, and cannot deal with categorical attributes as in our approach. Second, their work treats model parameters as fixed values, while we take a Bayesian treatment on the model parameters. Our Bayesian model essentially considers all possible parameter values and leads to more robust clustering results. [4] applies the Latent Dirichlet Allocation (LDA) to graph clustering. The proposed method is based on a different Bayesian framework from ours and it deals with edge attributes rather than vertex attributes.

In literature, the term "attributed graph clustering" sometimes refers to another clustering problem, in which a set of attributed graphs is given and the task is to group the graphs into clusters [14, 13, 15, 8, 6]. Their problem is different from ours since the objects to be clustered there are small *graphs*, while those to be clustered in our problem are the *vertices* in a single attributed graph.

7. CONCLUSIONS

We studied the problem of clustering attributed graphs. Unlike the existing works that define artificial distance measures to fuse the structural and attribute information, we proposed a natural and principled model-based approach to attributed graph clustering. We devised a Bayesian model to seamlessly leverage the structural and

attribute information in clustering an attributed graph, and transformed the clustering problem into a standard probabilistic inference problem. We then developed an efficient variational algorithm to solve the probabilistic inference problem.

Our experiments on real-world attributed graphs verified both the effectiveness and efficiency of our method. First, the experimental results show that our algorithm attains high clustering quality both structure-wise and attribute-wise, both of which being significantly superior to the currently best known algorithm for this task, Inc-Cluster [20]. Second, our algorithm is up to two orders of magnitude faster and consumes substantially less memory than Inc-Cluster. The results particularly show that our algorithm is far more scalable in clustering large attributed graphs than Inc-Cluster.

Model-based clustering methods are commonly considered slow and hard to scale up. Nonetheless, this work shows that our model-based method is far more efficient than the state-of-the-art distance-based method. Given the promising results, we hope that our work can attract more attention to model-based methods and stimulate the development in this line for large-scale data mining.

8. ACKNOWLEDGMENTS

This research is supported in part by the A*STAR Thematic Strategic Research Programme Grant (102 158 0034), the AcRF Tier-1 Grant (M52020092) and the MoE Grant (MOE2010-T2-2-071) from Ministry of Education of Singapore, and the grant of the Research Grants Council of the Hong Kong SAR, No. 411211.

9. REFERENCES

[1] H. Cheng, Y. Zhou, and J. X. Yu. Clustering large attributed graphs: A balance between structural and attribute similarities. *TKDD*, 5(2):12, 2011.

[2] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley-Interscience, 1991.

[3] M. H. DeGroot. *Proability and Statistics*. Addison-Wesley, 2nd edition, 1986.

[4] K. Henderson, T. Eliassi-Rad, S. Papadimitriou, and C. Faloutsos. Hcdf: A hybrid community discovery framework. In *SDM*, pages 754–765, 2010.

[5] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, 1999.

[6] B. J. Jain and F. Wysotzki. Central clustering of attributed graphs. *Machine Learning*, 56(1-3):169–207, 2004.

[7] M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[8] M. A. Lozano and F. Escolano. ACM: Attributed graph clustering for learning classes of images. In *Graph Based Representations in Pattern Recognition*, pages 247–258, 2003.

[9] J. Neville, M. Adler, and D. Jensen. Clustering relational data using attribute and link information. In *Text Mining and Link Analysis Workshop, IJCAI*, pages 689–698, 2003.

[10] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69:066133, 2004.

[11] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:066113, 2004.

[12] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

[13] A. Sanfeliu, R. Alquézar, and F. Serratosa. Clustering of attributed graphs and unsupervised synthesis of function-described graphs. In *International Conference on Pattern Recognition*, pages 6022–6025, 2000.

[14] D. S. Seong, H. S. Kim, and K. H. Park. Incremental clustering of attributed graphs. *IEEE Transactions on Systems, Man and Cybernetics*, 23(5):1399 –1411, 1993.

[15] F. Serratosa, R. Alquézar, and A. Sanfeliu. Synthesis of function-described graphs and clustering of attributed graphs. *International Journal of Pattern Recognition and Artificial Intelligence*, 16(6):621–656, 2002.

[16] K. Steinhaeuser and N. V. Chawla. Community detection in a large real-world social network. In *Social Computing, Behavioral Modeling, and Prediction*, pages 168–175. 2008.

[17] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *ICDM*, pages 613–622, 2006.

[18] H. Zanghi, S. Volant, and C. Ambroise. Clustering based on random graph model embedding vertex features. *Pattern Recognition Letters*, 31(9):830–836, 2010.

[19] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.

[20] Y. Zhou, H. Cheng, and J. X. Yu. Clustering large attributed graphs: An efficient incremental approach. In *ICDM*, pages 689–698, 2010.

APPENDIX

A. DERIVATION OF STATIONARY POINT EQUATIONS

Our starting point is the equation

$$\nabla \widetilde{L}(q) = \left(\frac{\partial \widetilde{L}}{\partial \tilde{\xi}}, \frac{\partial \widetilde{L}}{\partial \tilde{\gamma}}, \frac{\partial \widetilde{L}}{\partial \tilde{\mu}}, \frac{\partial \widetilde{L}}{\partial \tilde{\nu}}, \frac{\partial \widetilde{L}}{\partial \tilde{\beta}} \right) = 0.$$

We first simplify $\widetilde{L}(q)$ as follows:

$$\begin{aligned}
\tilde{L}(q) &= \sum_{\mathbf{Z}} \iiint q(\alpha, \theta, \phi, \mathbf{Z}) \log \frac{p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z} | \xi, \gamma, \mu, \nu)}{q(\alpha, \theta, \phi, \mathbf{Z})} \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi \\
&= E_{\tilde{\xi}}[\log p(\alpha)] + E_{\tilde{\gamma}}[\log p(\theta)] + E_{\tilde{\mu},\tilde{\nu}}[\log p(\phi)] + E_{\tilde{\xi},\tilde{\beta}}[\log p(\mathbf{Z}|\alpha)] \\
&\quad + E_{\tilde{\mu},\tilde{\nu},\tilde{\beta}}[\log p(\mathbf{X}|\mathbf{Z},\phi)] + E_{\tilde{\gamma},\tilde{\beta}}[\log p(\mathbf{Y}|\mathbf{Z},\theta)] - E_{\tilde{\xi}}[\log q(\alpha)] \\
&\quad - E_{\tilde{\gamma}}[\log q(\theta)] - E_{\tilde{\mu},\tilde{\nu}}[\log q(\phi)] - \sum_{i=1}^{N} E_{\tilde{\beta}_i}[\log q(Z_i)].
\end{aligned}$$

The expectations are taken with respect to the variational distribution $q(\cdot|\cdot)$ and the subscripts denote the variational parameters involved in the expressions. For example,

$$E_{\tilde{\xi}}[\log p(\alpha)] = \int q(\alpha|\tilde{\xi}) \log p(\alpha) \mathrm{d}\alpha.$$

We now derive the partial derivatives one by one, starting with $\frac{\partial \tilde{L}}{\partial \tilde{\xi}}$. For simplicity, we collect the terms involving $\tilde{\xi}$ in $\widetilde{L}(q)$, and denote the sum as

$$\begin{aligned}
\tilde{L}_{\tilde{\xi}} &= E_{\tilde{\xi}}[\log p(\alpha)] + E_{\tilde{\xi},\tilde{\beta}}[\log p(\mathbf{Z}|\alpha)] - E_{\tilde{\xi}}[\log q(\alpha)] \\
&= \log \left\{ \frac{\prod_{k=1}^{K} \log \Gamma\left(\tilde{\xi}_k\right)}{\Gamma\left(\sum_{k=1}^{K} \tilde{\xi}_k\right)} \right\} \\
&\quad + \sum_{k=1}^{K} \left(\xi_k - \tilde{\xi}_k + \sum_{i=1}^{N} \tilde{\beta}_{ik} \right) \left[\psi\left(\tilde{\xi}_k\right) - \psi\left(\sum_{k=1}^{K} \tilde{\xi}_k \right) \right]. \quad (22)
\end{aligned}$$

Thus, for all $k = 1, 2, \ldots, K$,

$$\begin{aligned}
\frac{\partial \tilde{L}}{\partial \tilde{\xi}_k} = \frac{\partial \tilde{L}_{\tilde{\xi}}}{\partial \tilde{\xi}_k} &= \left(\xi_k - \tilde{\xi}_k + \sum_{i=1}^{N} \tilde{\beta}_{ik} \right) \psi'\left(\tilde{\xi}_k\right) \\
&\quad - \sum_{l=1}^{K} \left(\xi_l - \tilde{\xi}_l + \sum_{i=1}^{N} \tilde{\beta}_{il} \right) \psi'\left(\sum_{l=1}^{K} \tilde{\xi}_l \right).
\end{aligned}$$

Setting $\frac{\partial \tilde{L}}{\partial \tilde{\xi}_k} = 0$ for all k, we arrive at Equation (15)

$$\tilde{\xi}_k = \xi_k + \sum_{i=1}^{N} \tilde{\beta}_{ik}.$$

The stationary point equations for the other variational parameters can be derived similarly. For $\tilde{\gamma}$, we have

$$\begin{aligned}
\tilde{L}_{\tilde{\gamma}} &= E_{\tilde{\gamma}}[\log p(\theta)] + E_{\tilde{\gamma},\tilde{\beta}}[\log p(\mathbf{Y}|\mathbf{Z},\theta)] - E_{\tilde{\gamma}}[\log q(\theta)] \\
&= \sum_{t=1}^{T}\sum_{k=1}^{K} \log\left\{\frac{\prod_{m=1}^{M^t}\Gamma\left(\tilde{\gamma}_{km}^t\right)}{\Gamma\left(\sum_{m=1}^{M^t}\tilde{\gamma}_{km}^t\right)}\right\} \\
&\quad + \sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{m=1}^{M^t}\left[\gamma_m^t - \tilde{\gamma}_{km}^t + \sum_{i=1}^{N}\tilde{\beta}_{ik}\delta\left(\mathbf{Y}_i^t, a_m^t\right)\right] \\
&\quad \left[\psi\left(\tilde{\gamma}_{km}^t\right) - \psi\left(\sum_{m=1}^{M^t}\tilde{\gamma}_{km}^t\right)\right].
\end{aligned}$$

Setting $\frac{\partial \tilde{L}}{\partial \tilde{\gamma}_{km}^t} = \frac{\partial \tilde{L}_{\tilde{\gamma}}}{\partial \tilde{\gamma}_{km}^t} = 0$ for all t, k, m, we obtain Equation (16)

$$\tilde{\gamma}_{km}^t = \gamma_m^t + \sum_{i=1}^{N}\tilde{\beta}_{ik}\delta\left(\mathbf{Y}_i^t, a_m^t\right).$$

For $\tilde{\mu}$ and $\tilde{\nu}$, we have

$$\begin{aligned}
\tilde{L}_{\tilde{\mu},\tilde{\nu}} &= E_{\tilde{\mu},\tilde{\nu}}[\log p(\phi)] + E_{\tilde{\mu},\tilde{\nu},\tilde{\beta}}[\log p(\mathbf{X}|\mathbf{Z},\phi)] - E_{\tilde{\mu},\tilde{\nu}}[\log q(\phi)] \\
&= \sum_{k\le l}\log\left\{\frac{\Gamma\left(\tilde{\mu}_{kl}\right)\Gamma\left(\tilde{\nu}_{kl}\right)}{\Gamma\left(\tilde{\mu}_{kl}+\tilde{\nu}_{kl}\right)}\right\} \\
&\quad + \sum_{k=1}^{K}\left[\mu - \tilde{\mu}_{kk} + \sum_{i<j}\tilde{\beta}_{ik}\tilde{\beta}_{jk}\mathbf{X}_{ij}\right]\left[\psi\left(\tilde{\mu}_{kk}\right) - \psi\left(\tilde{\mu_{kk}} + \tilde{\nu}_{kk}\right)\right] \\
&\quad + \sum_{k=1}^{K}\left[\nu - \tilde{\nu}_{kk} + \sum_{i<j}\tilde{\beta}_{ik}\tilde{\beta}_{jk}\left(1-\mathbf{X}_{ij}\right)\right]\left[\psi\left(\tilde{\nu}_{kl}\right) - \psi\left(\tilde{\mu}_{kl}+\tilde{\nu}_{kl}\right)\right] \\
&\quad + \sum_{k<l}\left[\mu - \tilde{\mu}_{kl} + \sum_{i\neq j}\tilde{\beta}_{ik}\tilde{\beta}_{jl}\mathbf{X}_{ij}\right]\left[\psi\left(\tilde{\mu}_{kl}\right) - \psi(\tilde{\mu_{kl}} + \tilde{\nu}_{kl})\right] \\
&\quad + \sum_{k<l}\left[\nu - \tilde{\nu}_{kl} + \sum_{i\neq j}\tilde{\beta}_{ik}\tilde{\beta}_{jl}\left(1-\mathbf{X}_{ij}\right)\right]\left[\psi\left(\tilde{\nu}_{kl}\right) - \psi(\tilde{\mu}_{kl}+\tilde{\nu}_{kl})\right].
\end{aligned}$$

Setting $\frac{\partial \tilde{L}_{\tilde{\mu},\tilde{\nu}}}{\partial \tilde{\mu}_{kl}} = \frac{\partial \tilde{L}_{\tilde{\mu},\tilde{\nu}}}{\partial \tilde{\nu}_{kl}} = 0$ for all $k \le l$, we obtain Equations (17)–(20)

$$\begin{aligned}
\tilde{\mu}_{kk} &= \mu + \sum_{i<j}\tilde{\beta}_{ik}\tilde{\beta}_{jk}\mathbf{X}_{ij}, \\
\tilde{\nu}_{kk} &= \nu + \sum_{i<j}\tilde{\beta}_{ik}\tilde{\beta}_{jk}\left(1-\mathbf{X}_{ij}\right), \\
\tilde{\mu}_{kl} &= \mu + \sum_{i\neq j}\tilde{\beta}_{ik}\tilde{\beta}_{jl}\mathbf{X}_{ij}, \\
\tilde{\nu}_{kl} &= \nu + \sum_{i\neq j}\tilde{\beta}_{ik}\tilde{\beta}_{jl}\left(1-\mathbf{X}_{ij}\right).
\end{aligned}$$

For $\tilde{\beta}$, we have

$$\begin{aligned}
\tilde{L}_{\tilde{\beta}_i} &= \sum_{k=1}^{K}\tilde{\beta}_{ik}\left[\psi\left(\tilde{\xi}_k\right) - \psi\left(\sum_{k=1}^{K}\tilde{\xi}_k\right)\right] - \sum_{k=1}^{K}\tilde{\beta}_{ik}\log\tilde{\beta}_{ik} \\
&\quad + \sum_{j\neq i}\sum_{k=1}^{K}\sum_{l=1}^{K}\tilde{\beta}_{ik}\tilde{\beta}_{jl}\mathbf{X}_{ij}\left[\psi\left(\tilde{\mu}_{kl}\right) - \psi\left(\tilde{\mu_{kl}} + \tilde{\nu}_{kl}\right)\right] \\
&\quad + \sum_{j\neq i}\sum_{k=1}^{K}\sum_{l=1}^{K}\tilde{\beta}_{ik}\tilde{\beta}_{jl}\left(1-\mathbf{X}_{ij}\right)\left[\psi\left(\tilde{\nu}_{kl}\right) - \psi\left(\tilde{\mu}_{kl}+\tilde{\nu}_{kl}\right)\right] \\
&\quad + \sum_{t=1}^{T}\sum_{k=1}^{K}\tilde{\beta}_{ik}\sum_{m=1}^{M^t}\delta(\mathbf{Y}_i^t, a_m^t)\left[\psi\left(\tilde{\gamma}_{km}^t\right) - \psi\left(\sum_{m=1}^{M^t}\tilde{\gamma}_{km}^t\right)\right],
\end{aligned}$$

subject to $\sum_{k=1}^{K}\tilde{\beta}_{ik} = 1$. Introducing Lagrange multiplier λ and then setting $\frac{\partial}{\partial \tilde{\beta}_{ik}}\left[\tilde{L}_{\tilde{\beta}_i} + \lambda(\sum_{k=1}^{K}\tilde{\beta}_{ik} - 1)\right] = 0$ for all i, k will lead to Equation (21)

$$\begin{aligned}
\tilde{\beta}_{ik} &\propto \exp\Bigg\{\left[\psi\left(\tilde{\xi}_k\right) - \psi\left(\sum_{k=1}^{K}\tilde{\xi}_k\right)\right] \\
&\quad + \sum_{j\neq i}\sum_{l=1}^{K}\tilde{\beta}_{jl}\mathbf{X}_{ij}\left[\psi\left(\tilde{\mu}_{kl}\right) - \psi\left(\tilde{\mu_{kl}} + \tilde{\nu}_{kl}\right)\right] \\
&\quad + \sum_{j\neq i}\sum_{l=1}^{K}\tilde{\beta}_{jl}\left(1-\mathbf{X}_{ij}\right)\left[\psi\left(\tilde{\nu}_{kl}\right) - \psi\left(\tilde{\mu}_{kl}+\tilde{\nu}_{kl}\right)\right] \\
&\quad + \sum_{t=1}^{T}\sum_{m=1}^{M^t}\delta(\mathbf{Y}_i^t, a_m^t)\left[\psi\left(\tilde{\gamma}_{km}^t\right) - \psi\left(\sum_{m=1}^{M^t}\tilde{\gamma}_{km}^t\right)\right]\Bigg\}.
\end{aligned}$$

B. PROOF SKETCH FOR PROPOSITION 1

As described in Algorithm 2, $q^{(n+1)}$ is obtained from $q^{(n)}$ by a sequence of variational parameter updating:

1. Update $\tilde{\xi}^{(n+1)}, \tilde{\gamma}^{(n+1)}, \tilde{\mu}^{(n+1)}, \tilde{\nu}^{(n+1)}$ according to Equations (15)–(20);
2. Update $\tilde{\beta}^{(n+1)}$ according to Equation (21).

In order to prove $\widetilde{L}\left(q^{(n)}\right) \le \widetilde{L}\left(q^{(n+1)}\right)$, we only need to show that none of the updating will decrease the value of $\widetilde{L}$. Due to space limit, we only prove this for updating $\tilde{\xi}^{(n+1)}$ in the following. The other cases can be shown similarly.

As shown in Appendix A, the objective function $\widetilde{L}$ can be decomposes into two parts,

$$\widetilde{L} = \widetilde{L}_{\tilde{\xi}} + \mathcal{C}.$$

Here, $\widetilde{L}_{\tilde{\xi}}$ is a function of $\tilde{\xi}$ as defined in Equation (22), while $\mathcal{C}$ is a constant with respect to $\tilde{\xi}$. Because updating $\tilde{\xi}$ only affects $\widetilde{L}_{\tilde{\xi}}$, we only need to show $\widetilde{L}_{\tilde{\xi}^{(n)}} \le \widetilde{L}_{\tilde{\xi}^{(n+1)}}$.

Rearranging the expression of $\widetilde{L}_{\tilde{\xi}}$ in Equation (22), we get

$$\tilde{L}_{\tilde{\xi}} = -\mathrm{KL}\left(q(\alpha|\tilde{\xi})||p'(\alpha)\right) + \log C$$

where

$$p'(\alpha) = \frac{1}{C}\exp\left\{\log p(\alpha) + E_{\tilde{\beta}^{(n)}}[\log p(\mathbf{Z}|\alpha)]\right\},$$

$$C = \int \exp\left\{\log p(\alpha) + E_{\tilde{\beta}^{(n)}}[\log p(\mathbf{Z}|\alpha)]\right\}\mathrm{d}\alpha.$$

Note that $KL(q||p')$ is nonnegative and attains its minimum value zero if and only if $q = p$. Therefore, $\tilde{L}_{\tilde{\xi}}$ will be maximized at $\tilde{\xi}^\star$ that satisfies

$$q\left(\alpha|\tilde{\xi}^\star\right) = p'(\alpha).$$

Simplifying the above equation, we arrive at

$$\tilde{\xi}_k^\star = \xi_k + \sum_{i=1}^{N}\tilde{\beta}_{ik}^{(n)}, \quad k = 1, \cdots, K$$

Note this is exactly the updating equation for $\tilde{\xi}^{(n+1)}$. Consequently, we have

$$\tilde{L}_{\tilde{\xi}^{(n)}} \le \tilde{L}_{\tilde{\xi}^\star} = \tilde{L}_{\tilde{\xi}^{(n+1)}}.$$