GBAGC: A General Bayesian Framework for Attributed Graph Clustering

ZHIQIANG XU, Nanyang Technological University
YIPING KE and YI WANG, Institute of High Performance Computing
HONG CHENG and JAMES CHENG, The Chinese University of Hong Kong

Graph clustering, also known as community detection, is a long-standing problem in data mining. In recent years, with the proliferation of rich attribute information available for objects in real-world graphs, how to leverage not only structural but also attribute information for *clustering attributed graphs* becomes a new challenge. Most existing works took a distance-based approach. They proposed various distance measures to fuse structural and attribute information and then applied standard techniques for graph clustering based on these distance measures. In this article, we take an alternative view and propose a novel Bayesian framework for attributed graph clustering. Our framework provides a general and principled solution to modeling both the structural and the attribute aspects of a graph. It avoids the artificial design of a distance measure in existing methods and, furthermore, can seamlessly handle graphs with different types of edges and vertex attributes. We develop an efficient variational method for graph clustering under this framework and derive two concrete algorithms for clustering unweighted and weighted attributed graphs. Experimental results on large real-world datasets show that our algorithms significantly outperform the state-of-the-art distance-based method, in terms of both effectiveness and efficiency.

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

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Attributed graph clustering, model-based clustering, Bayesian method

ACM Reference Format:
Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. 2014. GBAGC: A general Bayesian framework for attributed graph clustering. ACM Trans. Knowl. Discov. Data 9, 1, Article 5 (August 2014), 43 pages.
DOI: http://dx.doi.org/10.1145/2629616

1. INTRODUCTION

Graph clustering is a fundamental data-mining and machine-learning technique for studying and understanding graph data. Classic graph clustering groups vertices in

This work is partially supported by the Hong Kong Research Grants Council (RGC) General Research Fund (GRF) Project No. CUHK 411211, 411310, and the Chinese University of Hong Kong Direct Grants No. 4055015 and No. 4055017.
Authors' addresses: Z. Xu, School of Computer Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798; email: zxu1@e.ntu.edu.sg; Y. Ke and Y. Wang, Institute of High Performance Computing, 1 Fusionopolis Way, #16-16 Connexis North, Singapore 138632; email: {keyp, wangyi}@ihpc.a-star.edu.sg; H. Cheng, Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N. T., Hong Kong; email: hcheng@se.cuhk.edu.hk; J. Cheng, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N. T., Hong Kong; email: jcheng@cse.cuhk.edu.hk.

DOI: http://dx.doi.org/10.1145/2629616

a given graph based on simple vertex connections (i.e., edges) to identify densely connected subgraphs or communities. In recent years, technological advances have generated an abundant amount of rich information for real-world objects. As a result, in addition to edge connections, vertices in graphs are now commonly associated with a list of attributes that describe the characteristics and properties of the real-world objects represented by the vertices. This gives rise to a new type of graph, namely, *attributed graphs*, and hence the demand of new clustering techniques for attributed graphs, called *attributed graph clustering*.

Attributed graph clustering is useful in many application domains as it yields more informative results. We list a few examples as follows:

—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 or her personal profile, such as interests, gender, education, residency, and so forth. Clustering the users in a social network by considering both their social relationships and personal profiles is particularly useful for social networking sites in service/apps recommendation, user-targeted online advertising, and so on.
—In the 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, and so forth. User groups discovered by clustering the attributed communication network can be used to design effective group-oriented marketing strategies in order to mitigate customer churns for telecom operators.
—In the World Wide Web, a web graph is composed of web pages and hyperlinks. Each web page can be described by attributes such as URL/domain, elements, keywords, tags, and so forth. Web communities discovered by considering both hyperlinks and web page attributes are more informative than those identified solely based on hyperlinks, since the former also utilizes web page contents whose similarity may not be captured by linkages.

The benefits of utilizing attribute information in addition to linkage information for graph clustering are obvious; however, the problem of attributed graph clustering is significantly more difficult as the presence of attributes poses grave new challenges to the task. An attributed graph contains two completely different types of information, structural connections and attribute values. Classic graph clustering and object clustering techniques handle either one of the two types but not both. Consequently, the resultant clusterings may not only miss important information but also lead to inaccurate findings. Therefore, how to meaningfully combine and leverage these two types of information is an essential task for attributed graph clustering.

The difficulty in handling attributed graphs does not only lie in the difference in the information nature (i.e., structural connections and attribute values). Greater challenges are posed as different applications actually generate graphs with different types of vertex attributes and edges. Attributed graphs can be social networks with categorical vertex attributes (e.g., gender, education, residency) and unweighted edges (e.g., friend-of), communication networks with numeric vertex attributes (e.g., data usage, total call charge, SMS usage) and weighted edges (e.g., number of communications), or networks with other types of combinations. How to develop a general solution that works for a wide variety of attributed graphs is challenging.

There have been some works on attributed graph clustering [Zhou et al. 2009; Cheng et al. 2011; Zhou et al. 2010; Neville et al. 2003; Steinhaeuser and Chawla 2008]. They mainly adopt the *distance-based* approach, which requires one to deliberately design a distance measure between graph vertices 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 does not work well in general, while learning weights for specific graphs is very time consuming [Zhou et al. 2009; Cheng et al. 2011; Zhou et al. 2010]. Furthermore, a distance measure is usually tailored for attributed graphs with a particular type of edges and vertex attributes. Generalizing the measure to graphs with other edge and attribute types is a nontrivial task.

In this article, we take an alternative view and propose a *General Bayesian framework to Attributed Graph Clustering* (**GBAGC**). Unlike the existing distance-based approaches, our framework is *model based*, which defines a probabilistic model that fuses structural and attribute information in a *natural and principled* manner. The model-based approach frees us from the problems of the distance-based approaches. On one hand, it avoids the artificial design of a distance measure. On the other hand, it is a general framework that can be easily instantiated to cluster various attributed graphs with different types of edges and attributes.

Our framework is grounded on two fundamental assumptions in graph clustering research: (1) there exists a *true but unknown* clustering of the vertices underlying the observed graph, and (2) vertices from the same cluster behave similarly to each other, while vertices from different clusters can behave differently. Under our framework, the cluster label of each vertex is explicitly represented as a *hidden* variable. Moreover, the intracluster similarity is enforced by assuming 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 a common set of distributions that are attached to that cluster. Via the hidden cluster variable, the structural and the attribute information of the vertices are naturally fused for attributed graph clustering.

Our framework essentially defines a joint probability distribution over the space of all possible attributed graphs and all possible clusterings of the vertices. For a given attributed graph to be clustered, the distribution induces a probability for each possible clustering. Therefore, the problem of attributed graph clustering can be transformed into a standard *probabilistic inference* problem, that is, to find the clustering that gives the highest probability [Pearl 1988]. Intuitively, this clustering best explains the observed attribute values and edge connections of the graph. Despite its conceptual simplicity, the inference problem is computationally intractable. To address this issue, we develop an efficient approximate solution based on the variational principle [Jordan et al. 1999]. We show that this solution is scalable on large graphs.

We summarize the main contributions of this article as follows:

—We propose a novel Bayesian framework for attributed graph clustering. Our framework avoids the artificial design of distance measures and provides a natural and principled way to leverage both structural and attribute information for clustering. Unlike existing works, which are designed to cluster specific types of attributed graphs, the new framework establishes a general approach to handle attributed graphs with a wide range of edge and attribute types.
—We show that the clustering problem can be transformed into a probabilistic inference task under the proposed framework. We analyze the computational hardness of the inference task and develop an efficient variational method for clustering.
—We demonstrate how the general framework can be easily instantiated to cluster various types of attributed graphs. We provide a practical guideline on the instantiation and exemplify it using two common types of attributed graphs, that is, unweighted and weighted graphs with categorical attributes. We also provide some insights on the applicability of the framework to other types of attributed graphs.

—We provide empirical evidence showing that our algorithm consistently outperforms the state-of-the-art distance-based algorithm [Zhou et al. 2010] on real-world attributed graphs. Our algorithm achieves significantly higher clustering quality in both structural and attribute aspects. Our algorithm is also drastically faster, from a few times on a small dataset to three orders of magnitude in larger datasets, and consumes substantially less memory. As a reference, we also compare our algorithm with a recent model-based attributed graph clustering method called PICS [Akoglu et al. 2012], which is based on a very different principle and handles only unweighted graphs with binary attributes. We show that our algorithm produces appealing alternative clustering results with much shorter running time.

Article Organization. The rest of the article is organized as follows. In Section 2, we formally define the problem of attributed graph clustering. In Section 3, we present our Bayesian framework. We then develop a variational clustering algorithm in Section 4. In Section 5, we derive two instances under our general framework for unweighted and weighted attributed graphs and discuss the instantiations to other types of attributed graphs. We report experimental results in Section 6 and discuss related works in Section 7. Finally, we conclude the article and point out future directions in Section 8.

2. PROBLEM STATEMENT

An attributed graph is defined as a 5-tuple $G = \langle V, \Lambda, F, H, W \rangle$, where $V = \{v_1, v_2, \ldots, v_N\}$ is a set of N vertices, $\Lambda = \{a_1, a_2, \ldots, a_T\}$ is a set of T attributes attached to each vertex, $F = \{f_1, f_2, \ldots, f_T\}$ is a set of T functions with $f_t : V \to dom(a_t)$ assigning to each vertex an attribute value from the domain $dom(a_t)$ of a_t, H is an attribute attached to each pair of vertices, and $W : V \times V \to dom(H)$ is a function that assigns an attribute value from the domain $dom(H)$ to each vertex pair. Note that this definition of attributed graph can accommodate a variety of structural connections, for example, unweighted graphs with $dom(H) = \{0, 1\}$ and weighted graphs with $dom(H) = \mathcal{R}$. In this article, we focus on *undirected* attributed graphs by restricting $W(v_i, v_j) = W(v_j, v_i)$, while our method can be easily extended to handle directed attributed graphs by removing this constraint.

Given an attributed graph G and the number of clusters K, the problem of attributed graph clustering is to partition the vertices V into K disjoint subsets $V_1, V_2, \ldots, V_K$, that is, $V = \bigcup_{i=1}^{K} V_i$ and $V_i \cap V_j = \emptyset$ for any $i \neq j$, such that (1) the vertices within each cluster are densely or strongly connected, while the vertices from different clusters are sparsely or weakly connected, and (2) the vertices within each cluster possess similar attribute values, while the vertices from different clusters may possess dissimilar attribute values.

3. A GENERAL BAYESIAN FRAMEWORK FOR ATTRIBUTED GRAPH CLUSTERING

In this section, we present a General Bayesian framework for Attributed Graph Clustering (GBAGC). Given a set of vertices V, the framework essentially defines a joint probability distribution over the space of all possible attributed graphs *and* all possible partitions over V. To cluster a given attributed graph, it performs probabilistic inference by calculating the posterior probability of each possible clustering for the graph and returns the most probable clustering with the maximum probability. Intuitively, this clustering best explains the attribute values and edge patterns of the given graph.

The framework is general in the sense that it is not tailored to specific types of the vertex and edge attributes Λ and H and can be conceptually applied to a variety of attributed graphs. We will discuss how this framework can be instantiated to cluster concrete attribute graphs with particular vertex and edge attributes in Section 5.

ALGORITHM 1: Generate Clustered Attributed Graph

Input: V, Λ, H, K
Output: $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$

1. Choose $\alpha \sim \texttt{Dirichlet}(\xi)$
2. For each cluster $k \in \{1, 2, \ldots, K\}$:
   (a) Choose $\theta_k \sim p(\theta_k|\gamma)$
   (b) For each cluster $l \in \{k, k+1, \ldots, K\}$:
   - Choose $\phi_{kl} \sim p(\phi_{kl}|\kappa)$
   - Set $\phi_{lk} = \phi_{kl}$
3. For each vertex $v_i \in V$:
   (a) Choose $Z_i \sim \texttt{Multinomial}(\alpha)$
   (b) Choose $Y_i \sim p(Y_i|\theta_{Z_i})$
   (c) For each vertex $v_j \in V$ with $i < j$:
   - Choose $X_{ij} \sim p(X_{ij}|\phi_{Z_i Z_j})$

We start by introducing some basic notions and notations in Section 3.1. We then describe an intuitive process for generating attributed graphs in Section 3.2. This process essentially draws samples of attributed graphs from an underlying distribution. In Section 3.3, we analyze the assumptions made by the generative process, based on which we formally define the distribution. Finally, in Section 3.4, we present a conceptual way to cluster attributed graphs under this framework.

3.1. Notions and Notations

Suppose we are given a set of vertices V, the vertex and edge attributes Λ and H, and the number of clusters K.

—An *adjacency matrix* $\mathbf{X} = [X_{ij}]$ is an $N \times N$ random matrix. Each element X_{ij} is a random variable that takes value from $dom(H)$. It indicates the existence or connection strength of the edge between vertices v_i and v_j.
—An *attribute matrix* $\mathbf{Y} = [Y_{it}]$ is an $N \times T$ random matrix. Each element Y_{it} is a random variable that takes value from $dom(a_t)$. It 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 $\{1, 2, \ldots, K\}$. It 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 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

Algorithm 1 outlines a process for generating clustered attributed graphs. The process takes as input a set of vertices V, a set of vertex attributes Λ, an edge attribute H, and the number of clusters K. It outputs a sample $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ from all possible clustered attributed graphs.

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) the adjacency matrix $\mathbf{X} = [X_{ij}]$, (2) the attribute matrix $\mathbf{Y} = [Y_{it}]$, and (3) the clustering of vertices $\mathbf{Z} = [Z_i]$. At Step 3 of Algorithm 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 choose α from a Bayesian prior.

(2) Given the cluster label Z_i of vertex v_i, we then sample the attribute values of this vertex (Step 3(b)). Specifically, we sample an attribute vector $Y_i = (Y_{i1}, \ldots, Y_{iT})$ from a distribution $p(Y_i|\theta_{Z_i})$. The distribution is parameterized by θ_{Z_i}. As indicated by its subscript, this parameter is specific to cluster Z_i. In other words, all vertices belonging to the same cluster share a common 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.

The parametric form of the distribution $p(Y_i|\theta_{Z_i})$ depends on the vertex attributes Λ. For example, it could be a multivariate normal distribution if the attributes in Λ take continuous values and are intercorrelated, or a product of T multinomial distributions if the attributes take categorical values and are mutually independent. We will leave it abstract when discussing the framework and come back to this issue in Section 5.

(3) Given the cluster labels Z_i and Z_j for vertex pair v_i and v_j, we finally sample the attribute value X_{ij} from a distribution $p(X_{ij}|\phi_{Z_iZ_j})$ (Step 3(c)), which denotes the existence or connection strength of the edge (v_i, v_j). The distribution is parameterized by $\phi_{Z_iZ_j}$. Note that this parameter depends on the cluster labels Z_i and Z_j. The implication is as follows. Suppose we are generating the samples X_{ik} and X_{jk} for two vertices v_i and v_j with respect to a common third vertex v_k. If v_i and v_j come from the same cluster, that is, $Z_i = Z_j$, we are sampling X_{ik} and X_{jk} from the same distribution. Otherwise, we are using different distributions for sampling. This is reasonable because vertices from the same cluster should be similar to each other, and they should have the same characteristic to interact with other vertices. On the other hand, for vertices from different clusters, the characteristic may diverge.

The parametric form of the distribution $p(X_{ij}|\phi_{Z_iZ_j})$ depends on the edge attribute H. For example, it could be a Bernoulli distribution if H takes value from $\{0, 1\}$ in the case of unweighted graphs or a normal distribution if H takes value from $\mathcal{R}$ in the case of weighted graphs. Again, we will leave it abstract until Section 5.

3.2.2. Generating α, θ, ϕ. To complete the generative process, we still need to specify the parameters α, θ, and ϕ. 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 *prior* distributions over them. We then sample their values from the prior distributions as follows. We will discuss the benefits of the Bayesian treatment in Section 3.4.

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

$$p(\alpha|\xi) = \frac{\Gamma\left(\sum_{k=1}^{K} \xi_k\right)}{\prod_{k=1}^{K} \Gamma(\xi_k)} \prod_{k=1}^{K} \alpha_k^{\xi_k - 1}, \tag{2}$$

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 *hyperparameter*[1] of the generative process in order to distinguish it from the parameter α.

The choice of the Dirichlet distribution is not arbitrary. It is mainly for mathematical convenience. Recall that we sample $\mathbf{Z}$ from the multinomial distribution parameterized by α. Mathematically, the Dirichlet is known as the *conjugate prior* for multinomial distribution [DeGroot 1986]. It means that, if the prior distribution of α is Dirichlet, then after we observe an instantiation of $\mathbf{Z}$, the posterior distribution of α is still Dirichlet. This will give rise to a closed-form expression for the posterior and facilitate the development of an efficient attributed graph clustering algorithm later. In the same spirit, we will enforce conjugate priors for θ and ϕ later.

(2) We place a conjugate distribution $p(\theta_k|\gamma)$ over θ_k for each cluster k and sample the value of θ_k from it (Step 2(a)). Note that the hyperparameter γ does not depend on the cluster label k. It is shared by all the K clusters. This should not be interpreted as that the parameters θ_k for different clusters are tied to the same value, and thus the attributes of the vertices from different clusters are drawn from the same distribution. In fact, at Step 2(a), we sample the parameter θ_k for each cluster k independently. Therefore, the values of θ_k vary across different clusters. This allows us to accommodate the intercluster heterogeneity of vertex attributes.

(3) Finally, we place a conjugate distribution $p(\phi_{kl}|\kappa)$ with hyperparameter κ over ϕ_{kl}, from which we sample the value of ϕ_{kl} for each cluster pair (k, l) independently (Step 2(b)). We set $\phi_{lk} = \phi_{kl}$ as the edges are undirected.

3.3. Bayesian Model for Clustered Attributed Graphs

In the previous subsection, we described a generative process for clustered attributed graphs. Each run of this process produces an instantiation of the parameters α, θ, ϕ and the clustered attributed graph $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$. 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 $\xi, \gamma, \kappa, \alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$. 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 these assumptions here, we give a compact graphical representation in Figure 1. Each node in the graph corresponds to one random variable. The shaded node represents either an observable or a hyperparameter. Each rectangle denotes the repetition of the enclosed component, where the number of repetitions is indicated by the subscript of the rectangle. For example, the rectangle encompassing the nodes Y_i and Z_i means that there are N nodes $Y_1, Y_2, \ldots, Y_N$, each of which has two parent nodes θ and Z_i, and N nodes $Z_1, Z_2, \ldots, Z_N$, each of which has one parent

[1]For simplicity, we set the hyperparameters ξ, γ, κ at predefined values in this article. We do note that they can also be optimized by maximum likelihood estimate [Blei et al. 2001] for potentially improved performance.

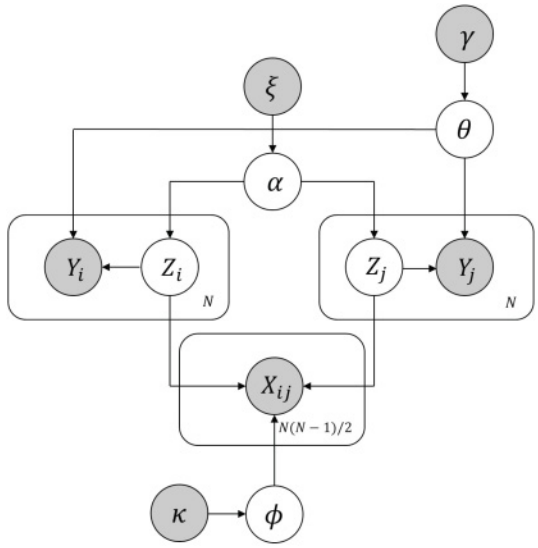


Fig. 1. A graphical representation of the proposed Bayesian model.

node α. The structure of the graph is constructed according to the flow of the generative process. Note that there is another rectangle that encompasses the nodes Y_j and Z_j with N repetitions. There are in fact only N nodes $Y_1, Y_2, \ldots, Y_N$ and N nodes $Z_1, Z_2, \ldots, Z_N$ in the model. This rectangle is duplicated to depict the dependency of X_{ij} on Z_i and Z_j.

The set of conditional independence assumptions can be readily read off from the graph. Specifically, a node is independent of all its nondescendants given its parent nodes [Pearl 1988]. For example, Z_i is independent of $\xi, \gamma, \kappa, \theta, \phi$ given α. This is because the generation of Z_i depends only on α (see Step 3(a) of Algorithm 1). Similarly, Z_i and Z_j are conditionally independent given α. This is because the cluster labels for different vertices are sampled independently.

Given the hyperparameters ξ, γ, κ, 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 model for clustered attributed graphs:

$$p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z}|\xi, \gamma, \kappa) = p(\alpha|\xi)p(\theta|\gamma)p(\phi|\kappa)p(\mathbf{Z}|\alpha)p(\mathbf{X}|\mathbf{Z}, \phi)p(\mathbf{Y}|\mathbf{Z}, \theta), \tag{3}$$

where

$$p(\theta|\gamma) = \prod_{k=1}^{K} p(\theta_k|\gamma), \tag{4}$$

$$p(\phi|\kappa) = \prod_{\substack{k,l=1 \\ k \le l}}^{K} p(\phi_{kl}|\kappa), \tag{5}$$

$$p(\mathbf{Z}|\alpha) = \prod_{i=1}^{N} p(Z_i|\alpha), \tag{6}$$

$$p(\mathbf{X}|\mathbf{Z}, \phi) = \prod_{\substack{i,j=1 \\ i<j}}^{N} p(X_{ij}|\phi_{Z_i Z_j}), \tag{7}$$

$$p(\mathbf{Y}|\mathbf{Z}, \theta) = \prod_{i=1}^{N} p(Y_i|\theta_{Z_i}), \tag{8}$$

and $p(Z_i|\alpha)$ and $p(\alpha|\xi)$ are defined in Equations (1) and (2), respectively. As discussed in Sections 3.2.1 and 3.2.2, we bear with the abstract forms of $p(\theta_k|\gamma)$, $p(\phi_{kl}|\kappa)$, $p(X_{ij}|\phi_{Z_i Z_j})$, and $p(Y_i|\theta_{Z_i})$ at the moment.

For brevity, we will omit the conditional part of the joint distribution $p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z}|\xi, \gamma, \kappa)$ and abbreviate it to $p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z})$ in the rest of this article. 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 hyperparameters ξ, γ, κ.

3.4. Attributed Graph Clustering as Probabilistic Inference

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 [Pearl 1988] of the clustering $\mathbf{Z}$ conditioning on $\mathbf{X}, \mathbf{Y}$:

$$\mathbf{Z}^\star = \arg\max_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \mathbf{Y}). \tag{9}$$

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.[2]

Despite its conceptual simplicity, the probabilistic inference problem is notoriously hard. There are two major difficulties. The first difficulty is that we need to jointly optimize the N variables $\mathbf{Z} = \{Z_1, Z_2, \ldots, Z_N\}$. The search space grows exponentially with the number of vertices N and is computationally prohibitive for large graphs. The second difficulty lies in the calculation of the posterior distribution of $\mathbf{Z}$. Note that

$$p(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) = \iiint p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y}) \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi, \tag{10}$$

where

$$p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y}) = \frac{p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z})}{\sum_{\mathbf{Z}} \iiint p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z}) \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi}. \tag{11}$$

Due to the integrals over the parameters α, θ, ϕ, there is no closed-form expression for $p(\mathbf{Z}|\mathbf{X}, \mathbf{Y})$. To address both difficulties, we develop an efficient algorithm to approximate $\mathbf{Z}^\star$ in the next section.

Before closing this section, we would like to emphasize the significance of the Bayesian treatment to the parameters α, θ, ϕ. This can be clearly seen from how we calculate the posterior $p(\mathbf{Z}|\mathbf{X}, \mathbf{Y})$ in Equation (10). By treating 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 the average over them by integration, rather than sticking to a single hypothetic value. In this way, the estimation of $p(\mathbf{Z}|\mathbf{X}, \mathbf{Y})$ is more reliable, which in turn leads to more robust clustering results.

[2] $\mathbf{Z}^\star$ gives a hard clustering of the graph vertices. Our framework can be used to produce soft clustering as well, where the degree that a vertex v_i belongs to cluster k is given by the probability $p(Z_i = k|\mathbf{X}, \mathbf{Y})$.

4. AN EFFICIENT VARIATIONAL ALGORITHM FOR CLUSTERING

4.1. Basic Idea

We now develop an efficient algorithm to approximate the optimal clustering $\mathbf{Z}^\star$. The algorithm is based on the variational principle described in Jordan et al. [1999]. The idea is to approximate the distribution $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$ defined in Equation (11) using a *variational distribution* $q(\alpha, \theta, \phi, \mathbf{Z})$ that is tractable for the integration over α, θ, ϕ and the maximization over $\mathbf{Z}$. We then approximate $\mathbf{Z}^\star$ as follows:

$$\begin{aligned}\mathbf{Z}^\star &= \arg\max_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) \\ &= \arg\max_{\mathbf{Z}} \iiint p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y}) \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi \\ &\approx \arg\max_{\mathbf{Z}} \iiint q(\alpha, \theta, \phi, \mathbf{Z}) \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi. \end{aligned} \tag{12}$$

Apparently, the success of this approximation depends on the choice of the variational distribution $q(\alpha, \theta, \phi, \mathbf{Z})$. On one hand, it should be amenable to efficient integration and maximization. On the other hand, it should be as close to the truth $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$ as possible. To this end, we restrict ourselves to a family of distributions that factorize as follows:

$$Q(\alpha, \theta, \phi, \mathbf{Z}) = Q(\alpha) Q(\theta) Q(\phi) \prod_i Q(Z_i). \tag{13}$$

We then choose the distribution within this family that is the closest to $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$ as the variational distribution $q(\alpha, \theta, \phi, \mathbf{Z})$. Due to this factorization, the integrals over α, θ, ϕ in Equation (12) diminish and the joint maximization over $\mathbf{Z}$ reduces to independent maximization over each Z_i,

$$\mathbf{Z}^\star \approx \left(\arg\max_{Z_1} q(Z_1), \arg\max_{Z_2} q(Z_2), \ldots, \arg\max_{Z_N} q(Z_N) \right). \tag{14}$$

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

4.2. Variational Family

In addition to the factorization assumption in Equation (13), we require the distributions in the variational family to be parametric and take the form

$$Q(\alpha, \theta, \phi, \mathbf{Z}|\tilde{\xi}, \tilde{\gamma}, \tilde{\kappa}, \tilde{\beta}) = Q(\alpha|\tilde{\xi}) Q(\theta|\tilde{\gamma}) Q(\phi|\tilde{\kappa}) \prod_i Q(Z_i|\tilde{\beta}_i), \tag{15}$$

where $\tilde{\xi}, \tilde{\gamma}, \tilde{\kappa}, \tilde{\beta}$ are the newly introduced *variational parameters*. This assumption is mild as we do not restrict the specific parametric forms of the marginals $Q(\alpha|\tilde{\xi})$, $Q(\theta|\tilde{\gamma})$, $Q(\phi|\tilde{\kappa})$, and $Q(Z_i|\tilde{\beta}_i)$. Rather, we allow them (and the variational parameters as well) to vary freely. As a consequence, the distributions in the variational family can now be exhausted by enumerating all possible parametric forms and variational parameters. Each instantiation specifies a unique distribution.

For simplicity, we will sometimes omit the conditional parts of the $Q(\cdot)$ distributions. For example, we will abbreviate $Q(\alpha, \theta, \phi, \mathbf{Z}|\tilde{\xi}, \tilde{\gamma}, \tilde{\kappa}, \tilde{\beta})$ 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.

4.3. Optimal Variational Distribution

Recall that our goal is to find the optimal variational distribution $q(\alpha, \theta, \phi, \mathbf{Z})$ in the family $Q(\alpha, \theta, \phi, \mathbf{Z})$ that is the closest to the true posterior $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$. Before we can seek the solution, we need to define a distance measure between the two distributions first to characterize the optimality. We adopt the Kullback-Leibler (KL) divergence [Cover and Thomas 1991] that is commonly used in information theory and machine learning. It is defined as

$$\mathrm{KL}(Q||p) = \sum_{\mathbf{Z}} \iiint Q(\alpha, \theta, \phi, \mathbf{Z}) \log \frac{Q(\alpha, \theta, \phi, \mathbf{Z})}{p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})} \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi. \tag{16}$$

Our problem is thus to find the variational distribution that minimizes the KL divergence

$$q(\alpha, \theta, \phi, \mathbf{Z}) = \arg\min_{Q} \mathrm{KL}(Q||p).$$

However, this optimization problem is intractable to solve 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

$$q(\alpha, \theta, \phi, \mathbf{Z}) = \arg\max_{Q} L(Q),$$

where

$$L(Q) = \sum_{\mathbf{Z}} \iiint Q(\alpha, \theta, \phi, \mathbf{Z}) \log \frac{p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z})}{Q(\alpha, \theta, \phi, \mathbf{Z})} \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi. \tag{17}$$

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

$$KL(Q||p) + L(Q) = \log p(\mathbf{X}, \mathbf{Y}). \tag{18}$$

Note that $L(Q)$ involves the joint $p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z})$ instead of the conditional $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$. The former can be efficiently evaluated according to Equation (3).

Our objective now is to maximize $L(Q)$ with respect to $Q(\alpha, \theta, \phi, \mathbf{Z})$. As we do not restrict the parametric forms of $Q(\alpha|\tilde{\xi})$, $Q(\theta|\tilde{\gamma})$, $Q(\phi|\tilde{\kappa})$, and $Q(Z_i|\tilde{\beta}_i)$, we need to determine the optimal forms as well as the optimal values of the variational parameters $\tilde{\xi}, \tilde{\gamma}, \tilde{\kappa}, \tilde{\beta}$. This appears to be a daunting task as there could be a large number of parametric forms. Fortunately, the following proposition ensures that the components $q(\alpha)$, $q(\theta)$, $q(\phi)$, and $q(Z_i)$ of the optimal variational distribution $q(\alpha, \theta, \phi, \mathbf{Z})$ must take the same parametric forms as the priors $p(\alpha)$, $p(\theta)$, $p(\phi)$, and $p(Z_i)$, respectively. As a result, we can safely restrict the parametric form of the variational family $Q(\alpha, \theta, \phi, \mathbf{Z})$ to be the same as these priors without missing the optimal solution. We only need to optimize the variational parameters, of which the stationary points are also characterized in the following proposition.

PROPOSITION 1. *The distributions $Q(\alpha)$, $Q(\theta)$, $Q(\phi)$, and $Q(Z_i)$ at the stationary points (including the maximum) of $L(Q)$ have exactly the same parametric forms as the prior distributions $p(\alpha)$, $p(\theta)$, $p(\phi)$, and $p(Z_i)$ and are characterized by the following equations:*

$$\tilde{\xi}_k = \xi_k + \sum_{i=1}^{N} \tilde{\beta}_{ik}, \tag{19}$$

$$Q(\theta_k) \propto \exp\left\{\log p(\theta_k) + \sum_{i=1}^{N} \tilde{\beta}_{ik} \log p(Y_i|\theta_k)\right\}, \tag{20}$$

$$Q(\phi_{kk}) \propto \exp\left\{\log p(\phi_{kk}) + \sum_{\substack{i,\, j=1 \\ i<j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jk} \log p(X_{ij}|\phi_{kk})\right\}, \tag{21}$$

$$Q(\phi_{kl}) \propto \exp\left\{\log p(\phi_{kl}) + \sum_{\substack{i,\, j=1 \\ i\neq j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jl} \log p(X_{ij}|\phi_{kl})\right\}, \tag{22}$$

$$\tilde{\beta}_{ik} \propto \exp\left\{E_{Q(\alpha)}[\log \alpha_k] + E_{Q(\theta_k)}[\log p(Y_i|\theta_k)] + \sum_{\substack{j=1 \\ j\neq i}}^{N} \sum_{l=1}^{K} \tilde{\beta}_{jl} E_{Q(\phi_{kl})}[\log p(X_{ij}|\phi_{kl})]\right\}, \tag{23}$$

for all $k, l = 1, \ldots, K, k < l$ *and* $i = 1, \ldots, N$.

The proof of Proposition 1 is mainly adapted from that of variational Bayesian EM [Beal 2003]. For clarity, we defer all the proofs and detailed derivations in this article to the appendices.

An important implication of Proposition 1 is that the optimal variational distribution $q(Z_i)$ will take the same form as $p(Z_i)$, that is, multinomial distribution. It is parameterized by the optimized $\tilde{\beta}_i = (\tilde{\beta}_{i1}, \tilde{\beta}_{i2}, \ldots, \tilde{\beta}_{iK})$. Plugging it back to Equation (14), an approximation to the optimal clustering $\mathbf{Z}^\star$ can now be easily obtained as

$$\mathbf{Z}^\star \approx \left(\arg\max_k \tilde{\beta}_{1k}, \arg\max_k \tilde{\beta}_{2k}, \ldots, \arg\max_k \tilde{\beta}_{Nk}\right). \tag{24}$$

4.4. Iterative Optimization Procedure

Based on Proposition 1 and Equation (24), we now present an iterative procedure (Algorithm 2) for attributed graph clustering. We call it *general Bayesian attributed graph clustering* (GBAGC). It takes as input an initial value $\tilde{\beta}^{(0)}$, a threshold ϵ, and a limit on the number of iterations $n_{\max}$. It returns an approximately optimal clustering $\mathbf{Z}^\star$ of the graph vertices.

Intuitively, Algorithm 2 starts with an initial guess of the variational parameter $\tilde{\beta}$ and iteratively improves the distribution $Q(\alpha, \theta, \phi, \mathbf{Z})$ by enforcing the stationary point conditions in Proposition 1. The following proposition shows that the objective function $L(Q)$ will keep increasing in each iteration.

PROPOSITION 2. *For all* $n = 1, 2, \ldots,$

$$L(Q^{(n-1)}) \leq L(Q^{(n)}).$$

According to Equation (18), the value of $L(Q)$ is bounded from above. Therefore, an immediate corollary of Proposition 2 is that Algorithm 2 is guaranteed to converge

ALGORITHM 2: GBAGC

Input: an initial value $\tilde{\beta}^{(0)}$, a threshold ϵ, a maximum number of iterations $n_{\max}$
Output: a vertex clustering $\mathbf{Z}^*$

1. $n \leftarrow 1$
2. Repeat
   (a) Given $\tilde{\beta}^{(n-1)}$, update $\tilde{\xi}^{(n)}$, $Q^{(n)}(\theta)$, $Q^{(n)}(\phi)$ according to Equations (19)–(22)
   (b) Given $\tilde{\xi}^{(n)}$, $Q^{(n)}(\theta)$, $Q^{(n)}(\phi)$, $\tilde{\beta}^{(n-1)}$, update $\tilde{\beta}^{(n)}$ according to Equation (23)
   (c) $n \leftarrow n+1$
   Until $L(Q^{(n)}) - L(Q^{(n-1)}) < \epsilon$ or $n > n_{\max}$
3. Return $\left(\arg\max_k \tilde{\beta}_{1k}^{(n)}, \arg\max_k \tilde{\beta}_{2k}^{(n)}, \ldots, \arg\max_k \tilde{\beta}_{Nk}^{(n)}\right)$

within a finite number of iterations. In particular, it will converge to a local maximum of $L(Q)$. Note that $L(Q)$ is nonconcave and can have multiple local maxima. Thus, the quality of the local maximum that Algorithm 2 converges to depends on the choice of the initial value $\tilde{\beta}^{(0)}$. We will discuss the initialization issue in the experiments.

4.5. Complexity Analysis

The time complexity of Algorithm 2 is dominated by the loop of Step 2. Each iteration of this loop updates the variational distributions $Q(\cdot)$ once according to Equations (19)–(23). Specifically, it updates $\tilde{\xi}$, $Q(\theta)$, $Q(\phi)$, and $\tilde{\beta}$ in sequence, which takes time $O(NK)$, $O(K(T_\theta + NT_Y))$, $O(K^2(T_\phi + N^2T_X))$, and $O(NK(T_{\tilde{\alpha}} + T_{\tilde{Y}} + NKT_{\tilde{X}}))$, respectively. Here, T_θ, T_Y, T_ϕ, and T_X denote the time required to evaluate the distributions $p(\theta_k)$, $p(Y_i|\theta_k)$, $p(\phi_{kl})$, and $p(X_{ij}|\phi_{kl})$, while $T_{\tilde{\alpha}}$, $T_{\tilde{Y}}$, and $T_{\tilde{X}}$ denote the time required to calculate the expectations $E_{Q(\alpha)}[\log \alpha_k]$, $E_{Q(\theta_k)}[\log p(Y_i|\theta_k)]$, and $E_{Q(\phi_{kl})}[\log p(X_{ij}|\phi_{kl})]$, respectively. Note that these quantities depend on the specific forms of the distributions $p(\cdot)$ and $Q(\cdot)$ but are constant in the number of vertices N and the number of clusters K. Therefore, the time complexity of one iteration of Algorithm 2 is $O(N^2K^2)$ and the total complexity of the algorithm is $O(N^2K^2I)$, where I is the number of iterations.

Despite the quadratic worst-case time complexity, Algorithm 2 can be substantially accelerated in practice. First of all, we note that real-world graphs usually exhibit sparse structures. This sparsity can be exploited to significantly reduce the time complexity. In particular, it often enables an efficient calculation of the summations over the adjacency matrix $\mathbf{X}$ in Equations (21)–(23), which are the root causes of the quadratic complexity. This intuition is formalized by the following proposition. Let $\mathrm{nnz}(\mathbf{X})$ be the number of nonzero entries in $\mathbf{X}$, or equivalently the number of graph edges. The proposition states that, under mild assumptions, a significant reduction in the time complexity from $O(N^2K^2I)$ down to $O(\mathrm{nnz}(\mathbf{X})K^2I)$ can be achieved.

PROPOSITION 3. *Assume that $p(X_{ij}|\phi_{kl})$ is from the exponential family [Casella and Berger 2001]. Let $s(X_{ij}) = (s_1(X_{ij}), \ldots, s_r(X_{ij}), \ldots, s_R(X_{ij}))^T$ denote the vector of sufficient statistics of $p(X_{ij}|\phi_{kl})$, and define $s_r(\mathbf{X})$ as the $N \times N$ square matrix consisting of elements $s_r(X_{ij})$, $i, j \in \{1, 2, \ldots, N\}$. If $\mathrm{nnz}(s_r(\mathbf{X})) = \mathrm{nnz}(\mathbf{X})$ for all $r \in \{1, 2, \ldots, R\}$, then each iteration of Algorithm 2 can be accomplished in $O(\mathrm{nnz}(\mathbf{X})K^2)$ time.*

We give the proof of Proposition 3 in Appendix C. Note that the assumptions of Proposition 3 are satisfied by an array of commonly used distributions for $p(X_{ij}|\phi_{kl})$. They include, but are not limited to, the Bernoulli distribution, binomial distribution, Possion distribution, exponential distribution, Weibull distribution, and Gaussian distribution. These distributions can be used to model a wide range of attributed graphs with different edge types. We give two concrete examples in Section 5.

Second, the convergence of Algorithm 2 can be speeded up by performing an asynchronous update. In each iteration of the algorithm, we alternate between two optimization steps. At Step 2(a), we update $\tilde{\xi}$, $Q(\theta)$, $Q(\phi)$ based on $\tilde{\beta}$. At Step 2(b), we update $\tilde{\beta}$ based on $\tilde{\xi}$, $Q(\theta)$, $Q(\phi)$. The key observation here is that the updating equations of $\tilde{\xi}$, $Q(\theta)$, $Q(\phi)$ depend only on $\tilde{\beta}$. Therefore, we choose to execute Step 2(b) multiple times in each iteration. This would increase the running time per iteration, but in return, it usually brings about more improvement in $L(Q)$ and leads to a much greater reduction in the number of iterations I toward convergence.

Finally, we note that Algorithm 2 is highly parallelizable. At Step 2(a), the updating of $\tilde{\xi}_k$, $Q(\theta_k)$, $Q(\phi_{kl})$ can be performed independently for different clusters k and l. Similarly, the updating of $\tilde{\beta}_{ik}$ at Step 2(b) can be performed independently for different vertexes i and clusters k. Therefore, a significant reduction in elapsed running time can be achieved by deploying parallel computing.

For space complexity, the main consumption comes from the storage of the adjacency matrix $\mathbf{X}$ and the attribute matrix $\mathbf{Y}$ of the input graph and the variational distributions, $Q(\cdot)$. Specifically, storing $\mathbf{X}$ and $\mathbf{Y}$ takes space $O(N^2)$ and $O(NT)$, respectively. Here, T is the number of vertex attributes. Storing $\tilde{\xi}$, $Q(\theta)$, $Q(\phi)$, and $\tilde{\beta}$ takes space $O(K)$, $O(KS_\theta)$, $O(K^2S_\phi)$, and $O(NK)$, respectively. Here, S_θ and S_ϕ denote the space required to keep the distributions $Q(\theta_k)$ and $Q(\phi_{kl})$. Again, they depend on the specific forms of the distributions but are constant in N and K. Since $T < N$ holds in general and $K \ll N$ holds for the clustering task, the total space complexity of Algorithm 2 is $O(N^2)$. The space consumption could be further reduced in practice by deploying sparse matrix representations.

5. APPLICATIONS TO SPECIFIC ATTRIBUTED GRAPHS

So far we have developed a general Bayesian framework for clustering attributed graphs. Throughout the development, we keep the edge and attribute types abstract and make no assumptions about them. In this section, we show how this framework can be easily instantiated to cluster specific graphs with particular types of edges and attributes.

We start by providing a general guideline here:

(1) Given a specific attributed graph to be clustered, identify the vertex attributes Λ and the edge attribute H.
(2) Choose appropriate parametric forms for the distributions $p(Y_i|\theta_{Z_i})$ and $p(X_{ij}|\phi_{Z_iZ_j})$ according to Λ and H.
(3) Choose the corresponding conjugate priors $p(\theta_k|\gamma)$ and $p(\phi_{kl}|\kappa)$.
(4) Plug all the distributions into Equations (19)–(23) and use Algorithm 2 to cluster the given graph.

In the next subsections, we will demonstrate how this guideline can be applied to handle two popular classes of attributed graphs. We will also shed some light on its applicability to other types of attributed graphs.

5.1. Unweighted Graph with Categorical Vertex Attributes

The attributed graphs with unweighted edges and categorical vertex attributes are arguably the most popular class in real-world applications. This class of attributed graphs can be naturally modeled under our framework by letting the edge attribute H be a binary variable that takes value from $\{0, 1\}$ and each vertex attribute $a_t \in \Lambda$ be a distinct categorical variable.

Accordingly, each element X_{ij} of the adjacency matrix is a binary random variable, and each element Y_{it} of the attribute matrix is a categorical random variable that takes value from the domain $dom(a_t)$. The standard choices of their distributions are

Bernoulli and multinomial, respectively, which are defined as follows:

$$p(X_{ij}|\phi_{Z_iZ_j}) = (1-\phi_{Z_iZ_j})^{1-X_{ij}}(\phi_{Z_iZ_j})^{X_{ij}}, \tag{25}$$

$$p(Y_i|\theta_{Z_i}) = \prod_{t=1}^{T} p(Y_{it}|\theta_{Z_it}), \tag{26}$$

$$p(Y_{it}|\theta_{Z_it}) = \prod_{m=1}^{M_t} \theta_{Z_itm}^{\delta(Y_{it},a_{tm})}. \tag{27}$$

Here, M_t is the cardinality of the domain $dom(a_t)$, a_{tm} is the mth value in $dom(a_t)$, and $\delta(Y_{it}, a_{tm})$ is the indicator function that takes value 1 if $Y_{it} = a_{tm}$ and 0 otherwise. The vector $\theta_{Z_it} = (\theta_{Z_it1}, \theta_{Z_it2}, \ldots, \theta_{Z_itM_t})$ is the parameter of the multinomial distribution for vertex attribute variable Y_{it}. The element θ_{Z_itm} denotes the proportion of vertices in cluster Z_i that take value a_{tm}. It satisfies $\theta_{Z_itm} \in [0, 1]$ and $\sum_{m=1}^{M_t} \theta_{Z_itm} = 1$.

The conjugate priors for the Bernoulli and multinomial distributions are Beta and Dirichlet, respectively. They are defined as

$$p(\phi_{kl}|\kappa) = \frac{\Gamma(\mu+\nu)}{\Gamma(\mu)\Gamma(\nu)}\phi_{kl}^{\mu-1}(1-\phi_{kl})^{\nu-1}, \tag{28}$$

$$p(\theta_k|\gamma) = \prod_{t=1}^{T} p(\theta_{kt}|\gamma_t), \tag{29}$$

$$p(\theta_{kt}|\gamma_t) = \frac{\Gamma\left(\sum_{m=1}^{M_t}\gamma_{tm}\right)}{\prod_{m=1}^{M_t}\Gamma(\gamma_{tm})}\prod_{m=1}^{M_t}\theta_{ktm}^{\gamma_{tm}-1}. \tag{30}$$

Here, $\kappa = (\mu, \nu)$ and $\gamma = (\gamma_1, \ldots, \gamma_T)$ are the hyperparameters.[3] Note that there is one dedicated hyperparameter γ_t for each attribute a_t. This is because the domains of different attributes are different.

Plugging Equations (25)–(30) into Equations (19)–(23), we arrive at the following updating rules:

$$\tilde{\gamma}_{ktm} = \gamma_{tm} + \sum_{i=1}^{N} \tilde{\beta}_{ik}\delta(Y_{it}, a_{tm}),$$

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

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

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

[3]Following the convention in Bayesian statistics, we set the values of ξ, κ, γ at 1. This leads to the *noninformative* priors [DeGroot 1986], which are equivalent to uniform distributions over α, ϕ, θ. They assign equal probabilities to all possible values of α, ϕ, θ. Intuitively, it reflects that our prior belief has no preference on any parameter value over the others.

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

$$\tilde{\beta}_{ik} \propto \exp\left\{\left[\psi(\tilde{\xi}_k) - \psi\left(\sum_{l=1}^{K}\tilde{\xi}_l\right)\right] + \sum_{t=1}^{T}\sum_{m=1}^{M_t}\delta\left(Y_{it}, a_{tm}\right)\left[\psi\left(\tilde{\gamma}_{ktm}\right) - \psi\left(\sum_{n=1}^{M_t}\tilde{\gamma}_{ktn}\right)\right] + \sum_{\substack{j=1 \\ j\neq i}}^{N}\sum_{l=1}^{K}\tilde{\beta}_{jl}\left[X_{ij}\psi(\tilde{\mu}_{kl}) + (1 - X_{ij})\psi(\tilde{\nu}_{kl}) - \psi(\tilde{\mu}_{kl} + \tilde{\nu}_{kl})\right]\right\},$$

for all $t = 1, \ldots, T, k, l = 1, \ldots, K, k < l$, and $i = 1, \ldots, N$. $\psi(\cdot)$ 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.

These specific rules can then be used by Algorithm 2 for clustering unweighted attributed graphs.

5.2. Weighted Graph with Categorical Vertex Attributes

Our general framework can be easily instantiated to deal with weighted graphs. In particular, we consider an interesting class of attributed graphs with nonnegative integer edge weights and categorical vertex attributes. Such weights can be used to represent connection strength between vertices, for example, the number of emails sent between two users in an email network, the number of calls made between two subscribers in a telecommunication network, the number of papers jointly published by two authors in a coauthor network, and so on.

The vertex attributes Λ, the distributions $p(Y_i|\theta_{Z_i})$, and the prior $p(\theta_{Z_i}|\gamma)$ remain the same as in the previous subsection since both consider categorical vertex attributes. The only change is with the edge attribute H (and thus X_{ij}). It is now a nonnegative random variable that takes value in $\{0, 1, 2, \ldots\}$. The standard distribution for this random variable is the Poisson:

$$p(X_{ij}|\phi_{Z_iZ_j}) = \exp\{-\phi_{Z_iZ_j}\}\frac{\phi_{Z_iZ_j}^{X_{ij}}}{X_{ij}!}, \tag{31}$$

where $\phi_{Z_iZ_j}$ denotes the average edge weight between cluster Z_i and cluster Z_j and satisfies $\phi_{Z_iZ_j} \in [0, \infty)$. The corresponding conjugate prior is the Gamma distribution, defined as

$$p(\phi_{kl}|\kappa) = \frac{\nu^{\mu}}{\Gamma(\mu)}\phi_{kl}^{\mu-1}\exp\{-\nu\phi_{kl}\}, \tag{32}$$

where $\kappa = (\mu, \nu)$ is the hyperparameter.[4] Plugging Equations (31) and (32) along with Equations (26), (27), (29), and (30) into Equations (19)–(23), we arrive at the following

[4]We set the values of μ, ν to be smaller than 1 so that the prior $p(\phi_{kl})$ provides as little information as possible about the true values of parameters ϕ_{kl} and let the data speak for themselves [Gelman 2006].

updating rules for weighted attributed graphs:

$$\tilde{\gamma}_{ktm} = \gamma_{tm} + \sum_{i=1}^{N} \tilde{\beta}_{ik}\delta\left(Y_{it}, a_{tm}\right),$$

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

$$\tilde{\nu}_{kk} = \nu + \sum_{\substack{i,\, j=1 \\ i<j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jk},$$

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

$$\tilde{\nu}_{kl} = \nu + \sum_{\substack{i,\, j=1 \\ i\neq j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jl},$$

$$\tilde{\beta}_{ik} \propto \exp\left\{\left[\psi(\tilde{\xi}_k) - \psi\left(\sum_{l=1}^{K}\tilde{\xi}_l\right)\right] + \sum_{t=1}^{T}\sum_{m=1}^{M_t}\delta\left(Y_{it}, a_{tm}\right)\left[\psi\left(\tilde{\gamma}_{ktm}\right) - \psi\left(\sum_{n=1}^{M_t}\tilde{\gamma}_{ktn}\right)\right] + \sum_{\substack{j=1 \\ j\neq i}}^{N}\sum_{l=1}^{K}\tilde{\beta}_{jl}\left[X_{ij}[\psi(\tilde{\mu}_{kl}) - \log\tilde{\nu}_{kl}] - \log(X_{ij}!) - \frac{\tilde{\mu}_{kl}}{\tilde{\nu}_{kl}}\right]\right\},$$

for all $t = 1, \ldots, T$, $k, l = 1, \ldots, K$, $k < l$, and $i = 1, \ldots, N$.

5.3. Other Types of Attributed Graphs

Besides the previous two examples, the GBAGC framework can be readily instantiated to attributed graphs with other types of edge connection and vertex attributes. For example, we can employ Gaussian distribution plus the Gaussian-Gamma prior to model edges weighted by real values. We can also use Poisson/Gaussian distributions to deal with integer/real-valued vertex attributes in a similar fashion. In fact, for almost all common types of edge and vertex attributes, there exist well-studied and tractable distributions and conjugate priors [Gelman et al. 2003], making GBAGC extensible to a variety of attributed graph clustering applications.

In this article, we have been focusing on homogeneous attributed graphs in which all the vertices/edges share the same attributes. It is worth mentioning that GBAGC can be readily generalized to deal with heterogeneous attributed graphs as well. This can be seen by noticing that the joint distribution over the vertex attributes and structural patterns factorizes (Equations (7) and (8)). Therefore, we are free to choose heterogeneous distributions and conjugate priors for vertices/edges with different attributes.

6. EXPERIMENTAL STUDY

To assess the effectiveness of our clustering framework, we evaluate the performance of two instances of the framework by comparing them with the state-of-the-art

Table I. Datasets

	# vertices	# edges	$\|dom(a_1)\|$	$\|dom(a_2)\|$	$dom(H)$
Political Blogs	1,490	16,715	2	NIL	{0, 1}
DBLP10K	10,000	27,867	3	100	{0, 1}
DBLP84K	84,170	201,334	3	100	{0, 1}
WtDBLP10K	10,000	27,867	3	100	[0, 59]
WtDBLP84K	84,170	201,334	3	100	[0, 59]

distance-based attributed graph clustering algorithm *Inc-Cluster* [Zhou et al. 2010] (an improved version of SA-Cluster [Zhou et al. 2009]) and a recent model-based algorithm *PICS* [Akoglu et al. 2012]. All the algorithms were implemented in Matlab and the experiments were run on machines with Linux OS, Intel Xeon 2.67GHz CPUs, and 12GB or 256GB of RAM.

6.1. Datasets

We use five real datasets in our experiments. Three of them are unweighted attributed graphs that were used in the evaluation of *Inc-Cluster* [Zhou et al. 2010]. The other two are the weighted version of two of the unweighted graphs, which we use to evaluate the weighted instance.

- **Political Blogs.** This dataset has 1,490 vertices and 19,090 edges. Each vertex represents a web blog on U.S. politics and each directed edge represents a hyperlink from one web blog to another. Each vertex is associated with an attribute, indicating the political leaning of the web blog, *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.** This dataset is a coauthor network extracted from the DBLP Bibliography data. Each vertex represents a scholar and each edge represents a coauthor 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 $\geq$20 publications, "prolific" for the scholars with $\geq$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 [Hofmann 1999] 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 coauthor network. It contains 84,170 scholars in 15 research fields. In addition to the four research fields used in *DBLP10K*, 11 additional 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*.
- **WtDBLP10K.** This dataset is the same as *DBLP10K* except that each edge carries a weight representing the number of papers jointly published by two scholars.
- **WtDBLP84K.** This dataset is the same as *DBLP84K* except that each edge carries a weight representing the number of papers jointly published by two scholars.

Table I summarizes the characteristics of the datasets, including the number of vertices, the number of edges (i.e., the number of vertex pairs that are of nonzero

weight), the domain size of each of the vertex attributes $|dom(a_i)|$, and the domain of the edge attribute $dom(H)$. Among the three unweighted datasets, the *Political Blogs* dataset is the smallest. The other two datasets are much larger, allowing us to test the scalability of the algorithms (e.g., *Inc-Cluster* consumes 60GB of memory for DBLP84K). The weighted version of the two larger datasets is used to evaluate weighted attributed graph clustering.

6.2. Experimental Settings

We give the details of the experimental settings in this subsection, including the algorithm for comparison, the measures that are used to assess the quality of a clustering, and the initialization of parameters in the algorithms.

6.2.1. Algorithms for Comparison. We compare our model-based clustering algorithm, **GBAGC**, with the state-of-the-art distance-based clustering algorithm, denoted by **Inc-Cluster** [Zhou et al. 2010]. In order to design a distance measure that considers both structure and attributes, Inc-Cluster constructs an augmented graph, which introduces an artificial vertex for each attribute value and links a vertex in the input graph G to the artificial vertex 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. For performance comparison on the weighted instance, we also modify Inc-Cluster to run on weighted graphs by utilizing the edge weights in the transition probability matrix for distance calculation.

To further demonstrate the benefits of incorporating the attribute information into clustering attributed graphs, we also compare GBAGC with the downgraded versions of itself and Inc-Cluster that use only the structure information for clustering. We denote these two alternatives as **GBAGC (structure only)** and **Inc-Cluster (structure only)**, respectively.

We also compare GBAGC with a recent model-based algorithm for attributed graph clustering, denoted by **PICS** [Akoglu et al. 2012]. PICS aims to find vertex groups with similar connectivity and homogeneous attributes. It casts the clustering problem as a data compression task and uses the minimum description length (MDL) principle to automatically choose the number of clusters. PICS only handles unweighted graphs with binary attributes. Therefore, we only test it on the first three datasets of unweighted graphs, and we transform each categorical attribute a with $|dom(a)|$ possible values to $|dom(a)|$ dummy binary attributes for PICS.

6.2.2. Clustering Quality Assessment. Since our objective is to cluster attributed graphs, we assess the quality of a clustering in two aspects: *structure* and *attribute*.

[**Structure Quality**] We use *modularity* as a quality measure for structure. Modularity [Newman and Girvan 2004] is popularly used in graph clustering to measure the strength of division of a graph into vertex clusters (also called communities).

To define modularity, we first introduce the following notions. Given a clustering $\{V_1, \ldots, V_K\}$, let E be the set of vertex pairs with nonzero weights (called edges), E_{kl} ($k \neq l$) the set of intercluster edges between V_k and V_l, and E_{kk} the set of intracluster edges in V_k. Then the fraction of intracluster edges in V_k is defined as $f_{kk} = \frac{|E_{kk}|}{|E|}$ and the fraction of intercluster 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 intracluster and intercluster 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

$$modularity\,(V_1, \ldots, V_K) = \sum_{k=1}^{K} \left(f_{kk} - \alpha_k^2 \right). \tag{33}$$

Intuitively, if edges were distributed at random, the expected fraction of intracluster edges in V_k is α_k^2. By subtracting this expected fraction (i.e., α_k^2) from the true fraction of intracluster edges in V_k (i.e., f_{kk}), modularity reflects the concentration of vertices within clusters compared to 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 intracluster edges exceeds the number expected on a random basis. Therefore, a clustering result with high modularity has dense connections among vertices within the same cluster and sparse connections among vertices across different clusters.

Note that modularity is also applicable to weighted networks [Newman 2004a]. Specifically, let $w_{kl} = \sum_{(v_i, v_j) \in E_{kl}} W(v_i, v_j)$ and $w = \sum_{(v_i, v_j) \in E} W(v_i, v_j)$, and define $f_{kk} = \frac{w_{kk}}{w}$ and $f_{kl} = f_{lk} = \frac{w_{kl}}{2w}$ $(k \neq l)$. Plugging them into Equation (33) gives the modularity for weighted graphs. Likewise, a higher modularity indicates a better clustering quality in the weighted case.

[**Attribute Quality**] For vertex attributes, we use *entropy* as a quality measure. Entropy is a well-accepted measure that quantifies the uncertainty of a random variable. In attributed graph clustering, entropy can be used to measure the degree of inconsistency of the attribute values in each cluster.

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

$$entropy\,(a_t, V_k) = -\sum_{m=1}^{|dom(a_t)|} p_{ktm} \log p_{ktm},$$

where p_{ktm} is the fraction of vertices in cluster V_k that take the mth value in $dom(a_t)$.

We then define the entropy of an attribute a_t with respect to the clustering $\{V_1, \ldots, V_K\}$ as

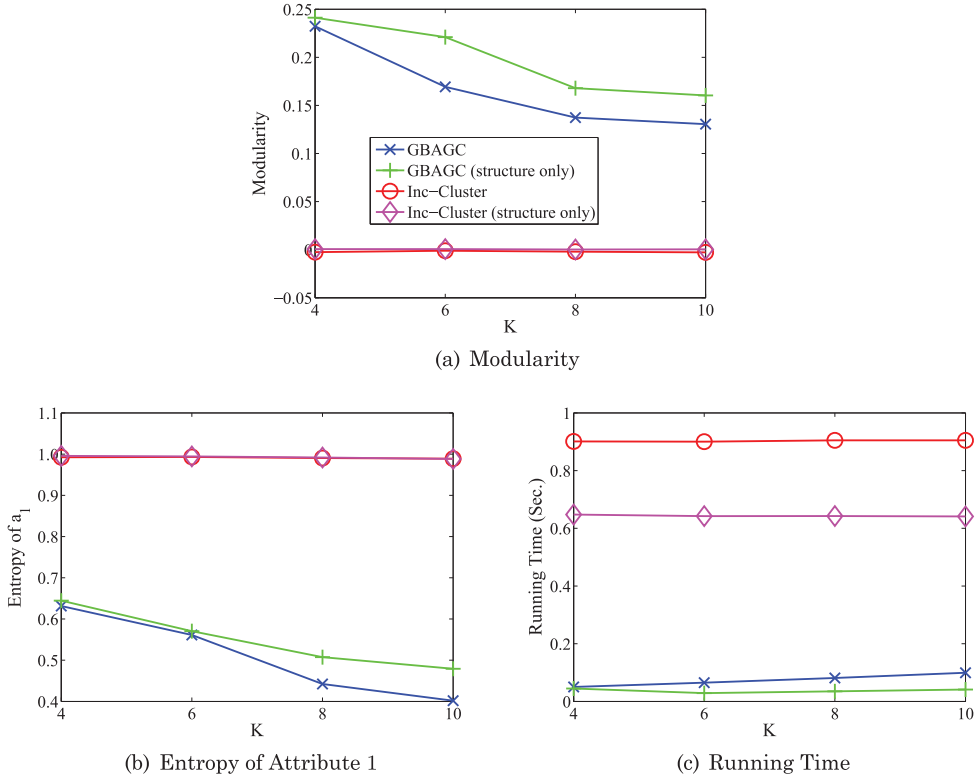
$$entropy\,(a_t) = \sum_{k=1}^{K} \frac{|V_k|}{|V|} entropy\,(a_t, V_k),$$

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 intracluster attribute similarity.

6.2.3. Parameter Settings and Implementations. We use METIS [Karypis and Kumar 1998] to initialize our GBAGC algorithm. METIS is a fast structure-based graph partitioning algorithm. It partitions the graph vertices into K equally sized clusters with minimum number or weighted sum of intercluster edges. To initialize GBAGC, we set $\tilde{\beta}_{ik} = 1$ if the ith vertex is assigned to the kth cluster by METIS and $\tilde{\beta}_{il} = 0$ for $l \neq k$.

We set all hyperparameters ξ, γ, and κ to 1 for experiments on unweighted graphs. For weighted graphs, the hyperparameter κ is set to 0.1 while ξ and γ are set to 1. The threshold ϵ for the objective function $L(Q)$ is set to 10^{-5}. The maximum number of iterations $n_{\max}$ is set to 10. We implement the speedup tricks for sparse graphs as shown in Proposition 3 and deploy sparse representations for matrices.

For Inc-Cluster, we use the default setting as in the code provided by the authors. For PICS, it is parameter free and we use the implementation by the authors.



(a) Modularity

(b) Entropy of Attribute 1

(c) Running Time

Fig. 2. Clustering performance of GBAGC and Inc-Cluster on Political Blogs.

6.3. Performance on Unweighted Attributed Graphs

In the following, we first report and discuss the performance of GBAGC and Inc-Cluster on the three datasets of unweighted graphs. We then compare their performance with that of PICS.

6.3.1. Clustering Performance on Political Blogs. For the Political Blogs dataset, we set the number of clusters at $K = 4, 6, 8,$ 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 GBAGC and Inc-Cluster. The result shows that, with respect to the modularity values (the higher the better), GBAGC achieves significantly higher-quality clustering than Inc-Cluster.

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

Next we assess the quality of clustering with respect to attribute information. Figure 2(b) reports the attribute entropy of the clustering obtained by GBAGC and Inc-Cluster for the Political Blogs dataset. GBAGC improves the attribute entropy of Inc-Cluster by 36% to 59%. The much lower entropy value of GBAGC shows that

GBAGC attains a much higher degree of consistency in intracluster 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 $L(Q)$ favors more skewed multinomial distribution and thus achieves a low attribute entropy in the clustering results. On the other hand, Inc-Cluster converts attribute values to artificial vertices and links them to original vertices to form an augmented graph. This increases the vertex connectivity with additional paths through the artificial vertices. 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 vertex of an attribute value a_{t1}, meaning that both u and v take the value a_{t1} of the attribute a_t. Then suppose that v is structurally connected to another vertex w and w takes a different attribute value a_{t2}. Inc-Cluster may put u, v, and w in the same cluster since their random walk scores can be high due to the paths through a_{t1}, even though u, v, and w do not exhibit high consistency on attribute a_t.

For different values of K tested, the clustering quality of Inc-Cluster is almost invariant in terms of both modularity and entropy. For GBAGC, the modularity degrades and the entropy improves when K increases. This reflects the natural tradeoff between the structural quality and attribute quality of a clustering solution. With more clusters and smaller average cluster size, it allows solutions with more consistent intracluster attribute values. On the other hand, the clusters would be more fragmental in this case and less indicative of inherent community structure. GBAGC finds a balance between these two aspects and achieves a stable overall clustering quality for different values of K.

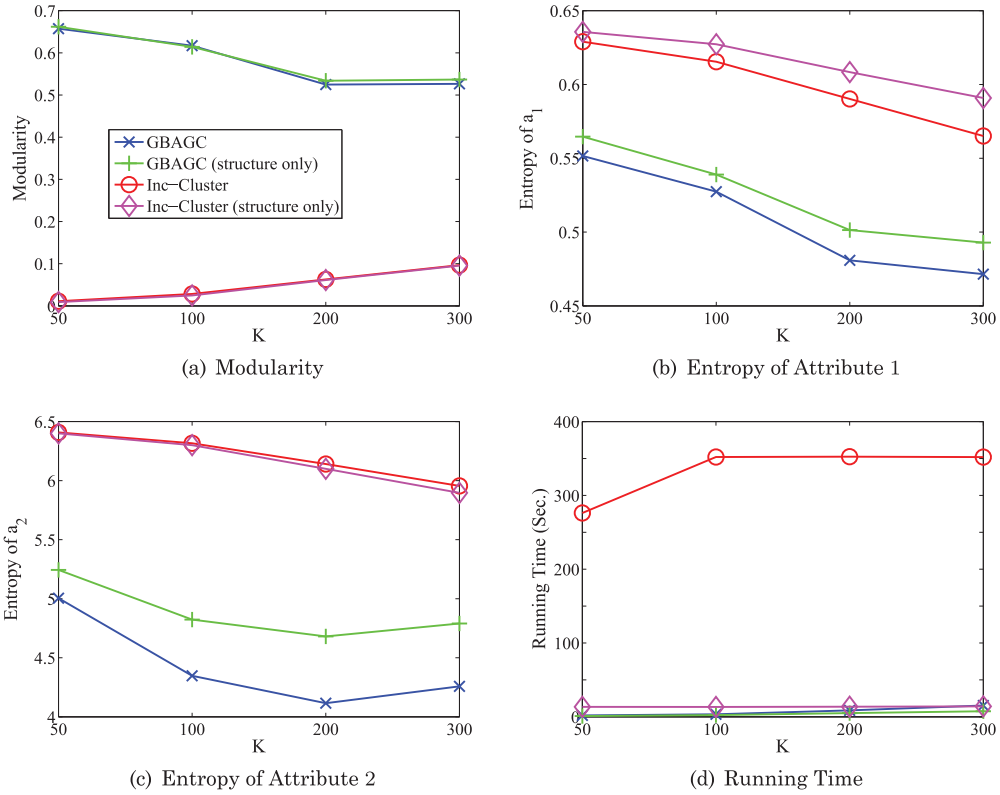
Figure 2(c) reports the running time of both algorithms for Political Blogs. On average, GBAGC is faster than Inc-Cluster by more than one order of magnitude. Therefore, in terms of both quality and efficiency, GBAGC is a clear winner for clustering this small dataset.

Figure 2 also reports the performance of the structure-only versions of GBAGC and Inc-Cluster. Comparing GBAGC with its structure-only version, we observe that it trades off modularity for entropy. It improves the attribute entropy over the structure-only version by 8% on average, but its modularity also decreases. This is not surprising because GBAGC explicitly enforces the intracluster attribute homogeneity, which may not always align with the structural homogeneity. In contrast, there are no significant changes in the performance of Inc-Cluster after incorporating the attribute information. Both the modularity and entropy are almost the same for Inc-Cluster and Inc-Cluster (structure only). If we further compare GBAGC with Inc-Cluster (structure only), we can see that the former exploits attribute information very effectively, resulting in an improvement in attribute entropy by 49% on average.

In terms of running time, the structure-only versions are faster since the incorporation of the attribute information into clustering incurs overhead. However, this overhead is much smaller in the case of GBAGC than in that of Inc-Cluster.

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

Figure 3(a) shows that GBAGC obtains high-quality clustering as the modularity is consistently over 0.5 for all values of K. According to Newman [2004b], a modularity value of 0.3 already indicates a significant community structure, that is, a high-quality clustering. On the contrary, Inc-Cluster records a low modularity of less than 0.1 for all values of K. The difference in the modularity value between GBAGC and Inc-Cluster is considerably greater in clustering this larger dataset than in clustering the smaller Political Blogs dataset.



(a) Modularity (b) Entropy of Attribute 1

(c) Entropy of Attribute 2 (d) Running Time

Fig. 3. Clustering performance of GBAGC and Inc-Cluster on DBLP10K.

Figures 3(b) and 3(c) report the entropy values of each of the two attributes[5] of DBLP10K. The results show that GBAGC attains considerably lower entropy values for both attributes than Inc-Cluster. On average, GBAGC improves the entropy of a_1 of Inc-Cluster by 15% and that of a_2 by 29%. This demonstrates the advantage of GBAGC over Inc-Cluster in attaining a higher-quality clustering with respect to attribute information, in addition to structural information.

When K increases, both modularity and attribute entropy of Inc-Cluster improve, but all of them are still significantly worse than those of GBAGC. For GBAGC, the trend is similar to that of Political Blogs. The entropy of both attributes improves in general as K increases, while modularity decreases. The only exception happens at $K = 300$, where GBAGC trades off the entropy of a_2 for the modularity and the entropy of a_1. In all cases, GBAGC produces high-quality clustering, as evidenced by much higher modularity and lower entropy than Inc-Cluster.

Figure 3(d) reports the running time of GBAGC and Inc-Cluster. The result shows that GBAGC is faster than Inc-Cluster by 23 to 147 times. As K increases, the running time of GBAGC increases linearly with K. However, for the running time of Inc-Cluster, we observe a different trend. There is a sudden increase in the running time of Inc-Cluster as K increases from 50 to 100, and then it uses almost the same amount of time in the range of K from 100 to 300. We examined the Inc-Cluster algorithm and found

[5]Note that the entropy value of a_1 is significantly lower than that of a_2 because the domain size of a_1 is significantly smaller than that of a_2, as shown in Table I.

that the number of iterations in its optimization process increases in a stepwise manner as K increases, and the running time is mainly determined by the number of iterations. Inc-Cluster uses four iterations for $K = 50$ and five iterations for $K \in [100..300]$, which explains the trend shown in Figure 3(d). More importantly, Figure 3(d) shows that the magnitude of increase in the running time of Inc-Cluster is significantly more rapid than the linear increase in the running time of GBAGC, indicating that GBAGC is also more scalable than Inc-Cluster as K increases.

When compared with their respective structure-only versions, GBAGC and Inc-Cluster behave differently. With the addition of attribute information, Inc-Cluster only improves the entropy of a_1, but the entropy of a_2 becomes worse. On the contrary, GBAGC achieves improvements over GBAGC (structure only) on the entropy of both attributes by 3% and 9%, respectively. Remarkably, the structure quality is not significantly compromised: the modularity merely decreases 1%. When compared with Inc-Cluster (structure only), GBAGC improves the entropy of a_1 and a_2 by 18% and 28%, respectively.

Again, the computational overhead paid by Inc-Cluster for incorporating the attribute information into clustering is significantly higher than that by GBAGC. The running time of GBAGC, GBAGC (structure only), and Inc-Cluster (structure only) is comparable to each other, while Inc-Cluster is up to two orders of magnitude slower.

6.3.3. 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 GBAGC for this largest dataset is the highest among the three datasets and is consistently over 0.6, indicating a very high clustering quality. Inc-Cluster produces low modularity of less than 0.1 for all values of K. The difference in the modularity value between GBAGC and Inc-Cluster also further widens, especially for small values of K.

Figures 4(b) and 4(c) further show that GBAGC consistently attains lower entropy values for both of the attributes than Inc-Cluster. GBAGC improves the entropy of a_1 and a_2 by 27% and 57%, respectively. Different from the results for DBLP10K, the entropy of Inc-Cluster for both attributes does not improve dramatically as K increases. On the contrary, GBAGC produces a clustering with much lower entropy for both attributes when K increases, at an expense of slightly worse modularity. This again demonstrates that GBAGC achieves a good tradeoff between structural and attribute quality as K changes.

Figure 4(d) shows a huge gap between the running time of GBAGC and that of Inc-Cluster. On average, GBAGC is faster than Inc-Cluster by over an order of magnitude. As explained for Figure 3(d) in Section 6.3.2, the running time of GBAGC increases linearly as K increases. On the contrary, the running time of Inc-Cluster increases in a stepwise manner.

On this large dataset, Inc-Cluster performs worse than Inc-Cluster (structure only) in all three measures, meaning that the way that Inc-Cluster exploits the attribute information actually hurts in this particular case. In contrast, GBAGC improves the entropy of a_1 and a_2 of GBAGC (structure only) by 3% and 20%, respectively, and meanwhile retains almost the same modularity level. GBAGC also gains about 20% and 31% improvement on the entropy of a_1 and a_2 over Inc-Cluster (structure only). Again, all these are achieved at a very small computational overhead, especially when compared to the extra cost incurred by Inc-Cluster.

6.3.4. Comparison with PICS. We now present the performance comparison of GBAGC, Inc-Cluster, and PICS. Since PICS automatically selects the cluster number K by itself, we set the K in GBAGC and Inc-Cluster to the same value for a fair comparison.

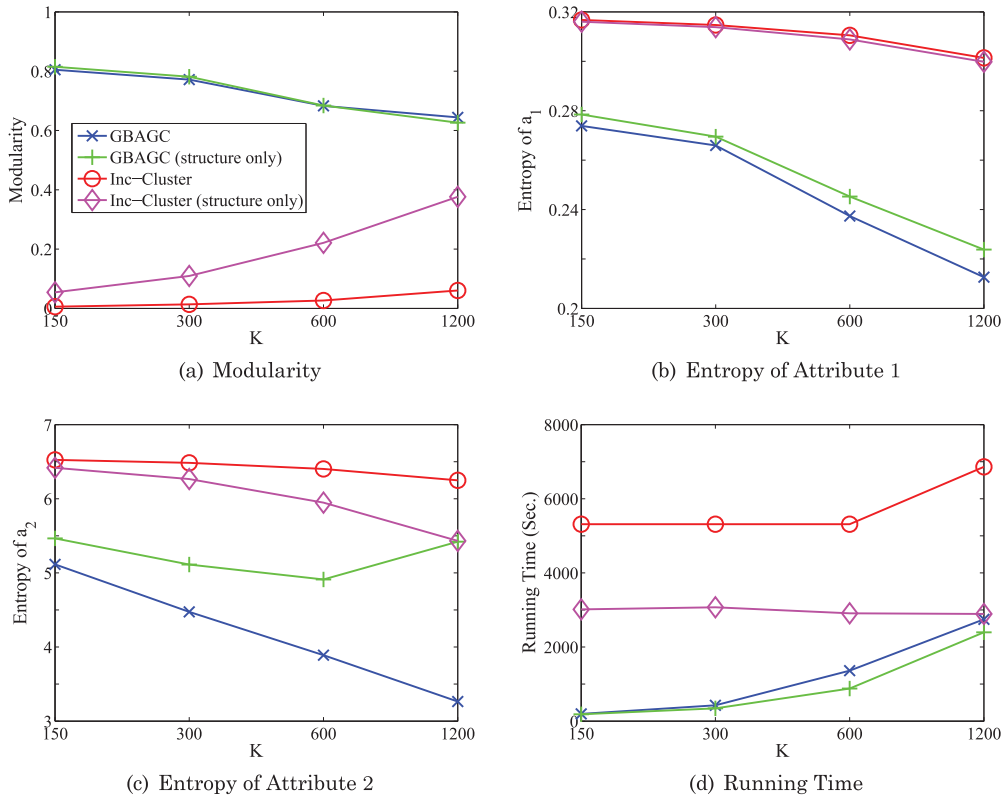


Fig. 4. Clustering performance of GBAGC and Inc-Cluster on DBLP84K.

Table II. Performance of GBAGC, Inc-Cluster, and PICS

Dataset	Algorithm	K	Modularity	Entropy of a_1	Entropy of a_2	Running Time (Sec.)
Political Blogs	GBAGC	11	0.133	**0.368**	-	**0.096**
	Inc-Cluster	11	−0.003	0.987	-	0.928
	PICS	11	**0.165**	0.572	-	36.477
DBLP10K	GBAGC	9	**0.655**	0.577	**5.925**	**0.239**
	Inc-Cluster	9	0.001	0.640	6.469	273.409
	PICS	9	0.337	**0.325**	6.275	251.554
DBLP84K	GBAGC	20	**0.825**	0.291	**5.876**	**18.160**
	Inc-Cluster	20	0.001	0.318	6.559	5484.222
	PICS	20	0.406	**0.136**	6.388	5308.746

Table II gives the results on the three datasets, where the best entry for each measure is highlighted in bold.

In terms of clustering quality, Inc-Cluster is the worst, while GBAGC and PICS achieve different tradeoffs between structure and attribute aspects. On political blogs, PICS obtains better modularity but GBAGC gets better attribute entropy. On both DBLP10K and DBLP84K, GBAGC generates clusterings with high modularity in structure and high homogeneity in attribute a_2, while PICS produces clusterings with high homogeneity in attribute a_1. On these two datasets, we argue that GBAGC produces more appealing clusterings since it induces more insights to identify researcher groups

Table III. Approximate Memory Usage

	Political Blogs	DBLP10K	DBLP84K
GBAGC	320MB	680MB	4GB
Inc-Cluster	460MB	4GB	60GB
GBAGC (K from PICS)	160MB	160MB	250MB
PICS	160MB	160MB	450MB

with tight collaborations (edge connections) and similar research topics (attribute a_2) than to identify groups with similar prolificacy in publication (attribute a_1).

In terms of running time, GBAGC is the clear winner. It is up to three orders of magnitude faster than Inc-Cluster and PICS. Although GBAGC and PICS share the linear time complexity in theory (in the number of graph edges), GBAGC is much more efficient in practice as shown in Table II.

6.3.5. Conclusions of Performance Comparison for Unweighted Attributed Graphs. In conclusion, the results in Sections 6.3.1 to 6.3.4 show that GBAGC consistently attains high-quality clustering in terms of both structural quality and attribute quality.

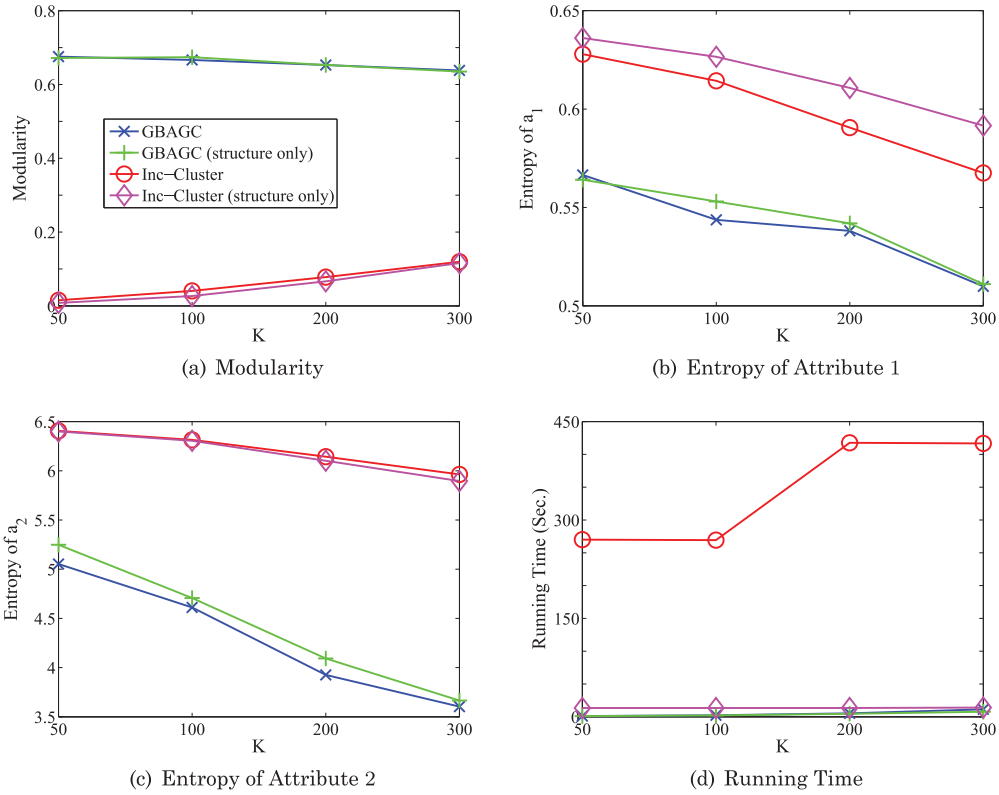
Compared with the state-of-the-art distance-based attributed graph clustering algorithm Inc-Cluster, the clustering obtained by GBAGC has significantly higher modularity and lower entropy for all datasets and all values of K tested. In addition, GBAGC is also consistently faster than Inc-Cluster, where the speedup in time is up to three orders of magnitude for clustering large datasets.

By comparing with the structure-only versions, we show that GBAGC is able to incorporate effectively and efficiently the attribute information into graph clustering. It leads to significant improvement in attribute quality, does not substantially compromise the structure quality, and incurs small computational overhead for the incorporation. In contrast, Inc-Cluster incurs a much larger cost, but the benefit of incorporating the attribute information turns out to be marginal.

We also show that GBAGC can produce competitive alternative clusterings to PICS, which takes a very different model-based approach to attributed graph clustering. GBAGC achieves a different tradeoff between structure and attribute quality from PICS. Particularly on DBLP datasets, we argue that GBAGC generates more meaningful clustering results. In addition, GBAGC has the same linear time complexity as PICS but is much more efficient to deploy in practice (faster by up to three orders of magnitude).

Finally, we briefly summarize the memory consumptions of the three algorithms. In general, GBAGC uses more memory with larger K value, while the memory consumption of Inc-Cluster remains unchanged across different K. As shown in Table III, even with the largest K values tested in our experiments, GBAGC (the first row) requires significantly less memory than Inc-Cluster. For DBLP84K, which is the largest among the three datasets, GBAGC consumed 4GB memory. In contrast, Inc-Cluster used 60GB, which made us have to run all the experiments for this dataset on a computer with 256GB of RAM. Under the same value of K, the memory consumption of GBAGC (the third row) is comparable to that of PICS on the two smaller datasets and is less than that of PICS on the large DBLP84K.

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.



(a) Modularity (b) Entropy of Attribute 1

(c) Entropy of Attribute 2 (d) Running Time

Fig. 5. Clustering performance on WtDBLP10K.

6.4. Performance on Weighted Attributed Graphs

We now report the performance of GBAGC and Inc-Cluster on the two weighted attributed graphs, WtDBLP10K and WtDBLP84K. We test the same values of K as in the unweighted case.

6.4.1. Clustering Performance on WtDBLP10K. Figure 5(a) reports the modularity of the clusterings obtained by the two algorithms. The modularity obtained by GBAGC is over 0.6 at all values of K, which is an indication of the detection of a very significant community structure. In contrast, the performance of Inc-Cluster is significantly worse where the modularity is very low. This demonstrates that GBAGC also achieves much higher structural quality than Inc-Cluster in clustering the weighted graph. Note that the modularity difference between GBAGC and Inc-Cluster in clustering graphs with the edge weight information is greater than that without the edge weight information, which shows that GBAGC is more flexible than Inc-Cluster in handling weighted attributed graphs.

Figures 5(b) and 5(c) report the entropy values for the two attributes of WtDBLP10K. GBAGC attains consistently lower entropy values than Inc-Cluster for both attributes. Figure 5(d) reports the running time. Similar to the unweighted case, GBAGC runs two orders of magnitude faster than Inc-Cluster on average. The running time of GBAGC increases very slowly with K, while that of Inc-Cluster exhibits a stepwise trend with a big increase in the steps.

On this dataset, both GBAGC and Inc-Cluster obtain similar modularity to their respective structure-only versions. In terms of attribute quality, Inc-Cluster achieves better entropy of a_1 but worse entropy of a_2 than Inc-Cluster (structure only). In contrast, GBAGC generally improves the entropy of both attributes by utilizing the attribute information. In terms of efficiency, it is the same as in the unweighted case. Inc-Cluster incurs much more computational cost than GBAGC in order to incorporate the attribute information into graph clustering.

It is interesting to compare the results here with those of DBLP10K. We observe that GBAGC brings about less improvement on the attribute entropy in this weighted case than in the unweighted one. This is mainly because the weights of the edges now carry extra information, some of which overlaps with the attribute information on the vertices. Specifically, each edge weight represents the number of papers coauthored by the two incident researchers. Such information is also indicative of the number of publications (attribute a_1) of each researcher and the similarity in the primary research topics (attribute a_2) of two researchers. Due to this correlation, the GBAGC (structure only) is able to use the weighted edges alone to generate clusters with relatively homogeneous attribute values. Adding the attribute information thus only improves the clustering quality slightly due to the effect of diminishing return. Taking a different perspective, this also demonstrates the capability of GBAGC in capturing the correlation between structure and attribute information through the common cluster label.

6.4.2. Clustering Performance on WtDBLP84K. The modularity values for WtDBLP84K are reported in Figure 6(a). For this large dataset, GBAGC also attains much higher modularity than Inc-Cluster. The modularity gap between GBAGC and Inc-Cluster is the biggest on WtDBLP84K among all the datasets for both unweighted and weighted cases. Figures 6(b) and 6(c) further show that GBAGC achieves significantly lower entropy than Inc-Cluster for both attributes. In addition, a huge gap in the running time between GBAGC and Inc-Cluster is again observed as shown in Figure 6(d).

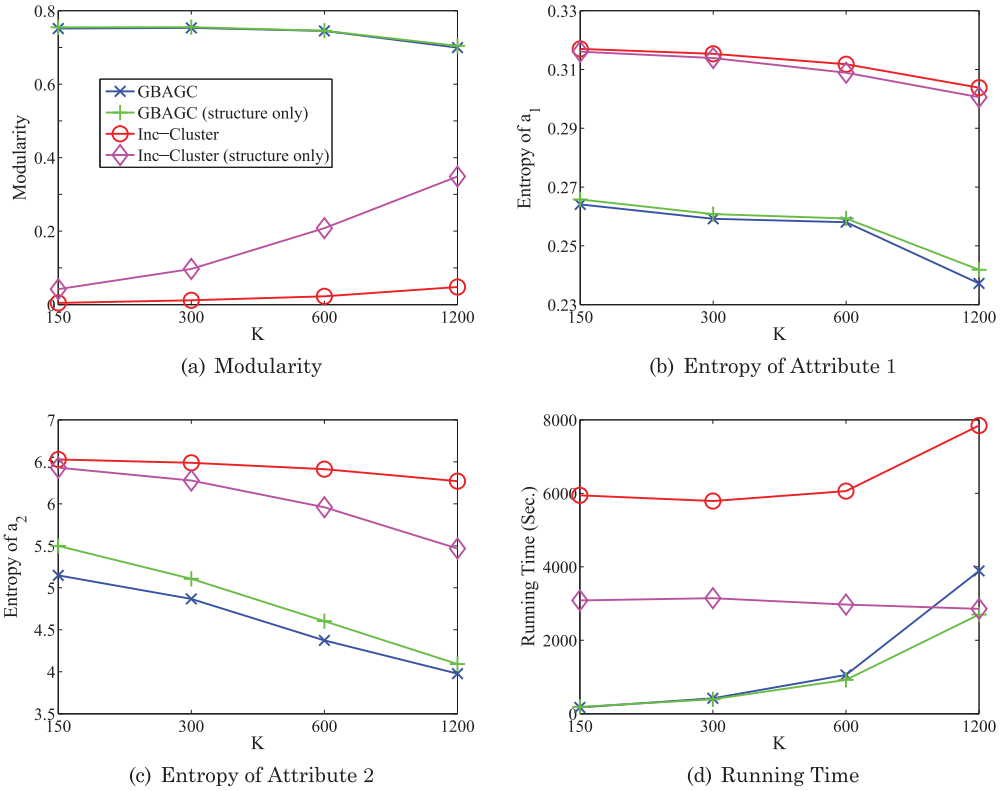
The comparison with structure-only versions on WtDBLP84K gives similar results to that on WtDBLP10K. The only difference is that Inc-Cluster obtains much worse modularity than Inc-Cluster (structure only), while GBAGC attains consistently high modularity in both versions.

6.4.3. Conclusions of Performance Comparison for Weighted Attributed Graphs. In conclusion, the experimental results of clustering WtDBLP10K and WtDBLP84K show that GBAGC significantly outperforms Inc-Cluster in terms of both efficiency and clustering quality for the weighted attributed graphs. Compared to the unweighted case, GBAGC demonstrates higher suitability in handling edge weight information than Inc-Cluster as their performance gap further widens in the weighted setting.

Compared with its structure-only version, GBAGC is able to improve the attribute quality in clustering weighted graphs when the attribute information is also considered. The improvement is smaller than that in the unweighted case due to the correlation between the edge weights and the attribute information. This in fact shows the capability of GBAGC in capturing such correlations embedded in data.

The memory usage of GBAGC and that of Inc-Cluster for the weighted graphs are comparable to those in the unweighted case as reported in Table III; that is, GBAGC also consumes significantly less memory than Inc-Cluster for the weighted attributed graphs.

The results demonstrate that our general model-based framework works well for weighted attributed graphs and the instantiation is superior to existing distance-based approaches, with much better clustering quality and higher clustering efficiency.



(a) Modularity (b) Entropy of Attribute 1

(c) Entropy of Attribute 2 (d) Running Time

Fig. 6. Clustering performance on WtDBLP84K.

7. RELATED WORK

There are a considerable number of works on attributed graph clustering in literature. They can be mainly categorized into two classes, *distance based* and *model based*.

The main idea of the distance-based approach 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. [2003] proposed a similarity measure for unweighted graphs with categorical attributes. The idea is to assign a weight to each edge using the attribute information. The weight is defined as the number of attribute values shared by the two end vertices. Standard graph clustering algorithms are then applied on the weighted adjacency matrix to perform clustering. This idea is extended by Steinhaeuser and Chawla [2008] to handle continuous attributes, where the differences in attribute values of neighboring vertices are used to define the edge weights. He et al. [2001] proposed a different similarity measure that is tailored to web page clustering. It combines the textual information and hyperlink structure. Based on this similarity measure, a hierarchical clustering algorithm is then applied to recursively partition the web pages.

The state-of-the-art distance-based algorithm is Inc-Cluster [Zhou et al. 2009, 2010]. It was designed for unweighted graphs with categorical attributes. It first constructs an augmented graph by creating an artificial vertex for each distinct attribute value and linking it to all the original vertices that possess this attribute value. A distance measure is then defined as the random walk score [Tong et al. 2006] computed

from the augmented graph. When computing the distance measure, Inc-Cluster automatically assigns different weights to structure and attributes. For the sake of efficiency, it computes the distances in an incremental fashion. Finally, the K-mediods algorithm is applied to cluster the vertices based on the distance measure. In this article, we compare our algorithm, GBAGC, with Inc-Cluster on unweighted graphs. We also modify Inc-Cluster to work for weighted graphs by utilizing the edge weights in the transition probability matrix for random walk score computation. As evidenced by the experimental results, our approach significantly outperforms Inc-Cluster in terms of both clustering quality and efficiency on both unweighted and weighted graphs.

Instead of artificially designing a distance measure, the model-based approach formulates a joint modeling of the interplay between edge connections and vertex attributes and makes use of this model to infer the optimal clustering. Zanghi et al. [2010] proposed a probabilistic model based on a similar generative process to ours. It can be treated as an instance of GBAGC for unweighted edges and continuous attributes, except that it does not take a Bayesian treatment but treats model parameters as fixed values. Henderson et al. [2010] studied attributed graph clustering in a different context. It aims at fusing clustering results from different algorithms to devise a hybrid approach. It treats the clustering results from other algorithms as categorical attributes on vertices. It then transforms these attributes to attributes on edges and applies LDA for graph clustering. Akoglu et al. [2012] proposed a different model-based approach named PICS to attributed graph clustering. It casts the clustering problem as a data compression task, where each cluster is treated as a compression of a cohesive subset of nodes that exhibit both similar connectivity patterns and high attribute homogeneity. It developed a matrix-based data compression model and applied the minimum description length (MDL) principle to find the optimal number of clusters and clustering. While the underlying principle can be extended to more general cases, the original paper only handles unweighted graphs with binary attributes. As shown in the experimental study, our approach and PICS achieve different tradeoffs between structure and attribute quality in clustering, but our approach is orders of magnitude faster than PICS. Recently, Xu et al. [2012] proposed a preliminary Bayesian method for attributed graph clustering. The method is developed for unweighted attributed graphs with categorical attributes and formed the basis of the current work.

As can be seen, most of the existing works on attributed graph clustering are tailored for specific types of edges and attributes. They cannot be trivially adapted to other data types. In contrast, our GBAGC establishes a general framework for attributed graph clustering with structured attributes. It inherits the idea of Xu et al. [2012] but extends the scope of applicability from merely unweighted graphs with categorical attributes to a much broader range of cases such as weighted edges and numerical attributes. To this end, the core operations that are shared when clustering different types of attributed graphs are abstracted, while the peripheral and type-dependent operations are articulated to make it easy for users to deal with specific attributed graphs at hand. To the best of our knowledge, this is the first attempt in literature that strives to promote the use of model-based graph clustering to this broader scope.

Our work is also remotely related to the works on community detection in the World Wide Web and citation networks. The goal is to identify subsets of documents or hypertexts that are densely connected and, meanwhile, contain similar textual contents or topics. The existing works usually leverage on sophisticated topic modeling techniques to incorporate the textual contents, such as probabilistic latent semantic analysis [Cohn

and Hofmann 2001; Nallapati et al. 2008] and LDA [Erosheva et al. 2004; Nallapati et al. 2008]. Yang et al. [2009a, 2009b] developed a different discriminative approach. It assumes that the cluster membership of each vertex follows a Gaussian process, which is defined by the textual contents. Recently, Sun et al. [2012] studied the problem of community detection in heterogeneous networks that contain different types of objects (e.g., articles, authors, publication venues, etc.) and connections (e.g., authorship, citation, etc.). It leverages on both the textual content of the articles and publication venues and the rich information in the heterogeneous links to perform clustering. All these works strive to take advantage of the unstructured text data to improve clustering quality. In contrast, our work focuses on structured attributes.

8. CONCLUSIONS

We studied the problem of clustering attributed graphs. Unlike the existing works that defined artificial distance measures to fuse the structural and attribute information, which are usually tailored for a specific type of attributed graphs, we proposed a natural and principled model-based approach for clustering various types of attributed graphs. More specifically, we devised a general Bayesian framework to seamlessly leverage the structural and attribute information in clustering generic attributed graphs. We then developed an efficient variational algorithm to solve the clustering problem under this framework. We also derived two instances from our general framework for clustering specific types of attributed graphs: unweighted and weighted ones.

Our experiments on real-world attributed graphs verified both the effectiveness and efficiency of our method. First, the experimental results show that the two instances of our GBAGC algorithm attain high clustering quality both structure-wise and attribute-wise and are significantly superior to the state-of-the-art algorithm for this task, Inc-Cluster [Zhou et al. 2010]. Second, our algorithms are up to three orders of magnitude faster and consume substantially less memory than Inc-Cluster. The results particularly show that our algorithms are more scalable in clustering large attributed graphs than Inc-Cluster. Third, comparing the unweighted and weighted instances, our algorithm is shown to be more superior to Inc-Cluster in fusing the extra edge weight information. This demonstrates the generality of our framework. Finally, our method provides a good alternative to existing model-based methods such as PICS [Akoglu et al. 2012], which is tailored for unweighted graphs with binary attributes and takes a much longer time to run in practice.

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 draw more attention to research on model-based methods and stimulate the development in this direction for large-scale data mining. In the future, we will further extend our framework to deal with more complicated and realistic datasets, such as missing value in attributes, heterogeneous types of edge connections and attribute values, and multiple edges.

APPENDIX

A. PROOF FOR PROPOSITION 1

Consider the following equality-constrained maximization problem:

$$\max_Q L(Q)$$
$$\text{subject to} \quad \sum_{\mathbf{Z}} \iiint Q(\alpha, \theta, \phi, \mathbf{Z}) \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi = 1.$$

Introducing a Lagrangian multiplier for each variational distribution, we obtain the Lagrange function:

$$\tilde{L}(Q) = L(Q) + \lambda_\alpha \left[\int Q(\alpha) \mathrm{d}\alpha - 1 \right] + \lambda_\theta \left[\int Q(\theta) \mathrm{d}\theta - 1 \right] + \lambda_\phi \left[\int Q(\phi) \mathrm{d}\phi - 1 \right] + \sum_i \lambda_i \left[\sum_{Z_i} Q(Z_i) - 1 \right],$$

where $L(Q)$ can be simplified as follows:

$$\begin{aligned} L(Q) &= \sum_{\mathbf{Z}} \iiint Q(\alpha, \theta, \phi, \mathbf{Z}) \log \frac{p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z})}{Q(\alpha, \theta, \phi, \mathbf{Z})} \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi \\ &= E_{Q(\alpha)}[\log p(\alpha)] + E_{Q(\alpha,\mathbf{Z})}[\log p(\mathbf{Z}|\alpha)] - E_{Q(\alpha)}[\log Q(\alpha)] \\ &\quad + E_{Q(\theta)}[\log p(\theta)] + E_{Q(\theta,\mathbf{Z})}[\log p(\mathbf{Y}|\mathbf{Z}, \theta)] - E_{Q(\theta)}[\log Q(\theta)] \\ &\quad + E_{Q(\phi)}[\log p(\phi)] + E_{Q(\phi,\mathbf{Z})}[\log p(\mathbf{X}|\mathbf{Z}, \phi)] - E_{Q(\phi)}[\log Q(\phi)] \\ &\quad - \sum_i E_{Q(Z_i)}[\log Q(Z_i)]. \end{aligned}$$

Note that the expectations in appendices are all taken with respect to variational distributions $Q(\cdot|\cdot)$ indicated by subscripts, for example,

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

Our starting point is the equation

$$\nabla \tilde{L}(Q) = \left(\frac{\partial \tilde{L}}{\partial Q(\alpha)}, \frac{\partial \tilde{L}}{\partial Q(\theta)}, \frac{\partial \tilde{L}}{\partial Q(\phi)}, \frac{\partial \tilde{L}}{\partial Q(Z_1)}, \ldots, \frac{\partial \tilde{L}}{\partial Q(Z_N)} \right) = 0.$$

We derive these partial derivatives one by one, starting with $\frac{\partial \tilde{L}}{\partial Q(\alpha)}$.

A.1. $\frac{\partial \tilde{L}}{\partial Q(\alpha)}$

For simplicity, we collect from $\tilde{L}(Q)$ the terms involving $Q(\alpha)$ in $\tilde{L}_{Q(\alpha)}$ and denote the sum as

$$\begin{aligned} \tilde{L}_{Q(\alpha)} &= E_{Q(\alpha)}[\log p(\alpha)] + E_{Q(\alpha,\mathbf{Z})}[\log p(\mathbf{Z}|\alpha)] - E_{Q(\alpha)}[\log Q(\alpha)] + \lambda_\alpha \int Q(\alpha) \mathrm{d}\alpha \\ &= \int Q(\alpha) \left[\log p(\alpha) + E_{Q(\mathbf{Z})}[\log p(\mathbf{Z}|\alpha)] - \log Q(\alpha) + \lambda_\alpha \right] \mathrm{d}\alpha. \end{aligned}$$

Thus,

$$\frac{\partial \tilde{L}}{\partial Q(\alpha)} = \frac{\partial \tilde{L}_{Q(\alpha)}}{\partial Q(\alpha)} = \log p(\alpha) + E_{Q(\mathbf{Z})}[\log p(\mathbf{Z}|\alpha)] - \log Q(\alpha) + \lambda_\alpha - 1.$$

Setting $\frac{\partial \tilde{L}}{\partial Q(\alpha)} = 0$, we arrive at

$$\begin{aligned} Q(\alpha) &= \exp\{-1 + \lambda_\alpha + \log p(\alpha) + E_{Q(\mathbf{Z})}[\log p(\mathbf{Z}|\alpha)]\} \\ &= \exp\{\log p(\alpha) + E_{Q(\mathbf{Z})}[\log p(\mathbf{Z}|\alpha)]\}/C_\alpha, \end{aligned}$$

where λ_α is related to the normalization constant C_α as follows:

$$1 - \lambda_\alpha = \log C_\alpha = \log \int \exp\{\log p(\alpha) + E_{Q(\mathbf{Z})}[\log p(\mathbf{Z}|\alpha)]\} \mathrm{d}\alpha.$$

The exponent can be further expanded as

$$\begin{aligned} \log p(\alpha) + E_{Q(\mathbf{Z})}[\log p(\mathbf{Z}|\alpha)] &= \sum_k (\xi_k - 1) \log \alpha_k + \sum_i E_{Q(Z_i)}[\log p(Z_i|\alpha)] \\ &= \sum_k \left\{ \xi_k - 1 + \sum_i E_{Q(Z_i)}[\delta(Z_i, k)] \right\} \log \alpha_k. \end{aligned}$$

Thus, $Q(\alpha|\tilde{\xi})$ is a Dirichlet distribution with parameters

$$\tilde{\xi}_k = \xi_k + \sum_i E_{Q(Z_i)}[\delta(Z_i, k)], \quad k = 1, 2, \ldots, K.$$

As we will see in Section A.4, variational distribution $Q(Z_i|\tilde{\beta}_i)$ is a multinomial distribution and thus $E_{Q(Z_i)}[\delta(Z_i, k)] = \tilde{\beta}_{ik}, k = 1, 2, \ldots, K$. Finally, we get Eq. (19):

$$\tilde{\xi}_k = \xi_k + \sum_i \tilde{\beta}_{ik}, \quad k = 1, 2, \ldots, K.$$

A.2. $\frac{\partial \tilde{L}}{\partial Q(\theta)}$

Collect from $\tilde{L}(Q)$ the terms involving $Q(\theta)$ in $\tilde{L}_{Q(\theta)}$:

$$\begin{aligned} \tilde{L}_{Q(\theta)} &= E_{Q(\theta)}[\log p(\theta)] + E_{Q(\theta,\mathbf{Z})}[\log p(\mathbf{Y}|\mathbf{Z}, \theta)] - E_{Q(\theta)}[\log Q(\theta)] + \lambda_\theta \int Q(\theta) \mathrm{d}\theta \\ &= \int Q(\theta) \left[\log p(\theta) + E_{Q(\mathbf{Z})}[\log p(\mathbf{Y}|\mathbf{Z}, \theta)] - \log Q(\theta) + \lambda_\theta \right] \mathrm{d}\theta. \end{aligned}$$

Setting $\frac{\partial \tilde{L}}{\partial Q(\theta)} = 0$, we arrive at

$$Q(\theta) = \exp\left\{ \log p(\theta) + E_{Q(\mathbf{Z})}[\log p(\mathbf{Y}|\mathbf{Z}, \theta)] \right\} / C_\theta,$$

where λ_θ is related to the normalization constant C_θ as follows:

$$1 - \lambda_\theta = \log C_\theta = \log \int \exp\left\{ \log p(\theta) + E_{Q(\mathbf{Z})}[\log p(\mathbf{Y}|\mathbf{Z}, \alpha)] \right\} \mathrm{d}\theta.$$

Since

$$\begin{aligned} \log p(\theta) &= \sum_k \log p(\theta_k), \\ E_{Q(\mathbf{Z})}[\log p(\mathbf{Y}|\mathbf{Z}, \theta)] &= \sum_i E_{Q(Z_i)}[\log p(Y_i|Z_i, \theta)], \\ E_{Q(Z_i)}[\log p(Y_i|Z_i, \theta)] &= \sum_k E_{Q(Z_i)}[\delta(Z_i, k)] \log p(Y_i|\theta_k) = \sum_k \tilde{\beta}_{ik} \log p(Y_i|\theta_k), \end{aligned}$$

the exponent can be further expanded as

$$\log p(\theta) + E_{Q(\mathbf{Z})}[\log p(\mathbf{Y}|\mathbf{Z}, \theta)] = \sum_k \left[\log p(\theta_k) + \sum_i \tilde{\beta}_{ik} \log p(Y_i|\theta_k) \right].$$

Thus, we have

$$Q(\theta|\tilde{\gamma}) = \prod_k Q(\theta_k|\tilde{\gamma}_k), \text{ and } Q(\theta_k|\tilde{\gamma}_k) = \exp\left\{ \log p(\theta_k) + \sum_i \tilde{\beta}_{ik} \log p(Y_i|\theta_k) \right\} \Big/ C_{\theta_k},$$

$k = 1, 2, \ldots, K$, where $C_{\theta_k} = \int \exp\{\log p(\theta_k) + \sum_i \tilde{\beta}_{ik} \log p(Y_i|\theta_k)\} \mathrm{d}\theta_k$. Equation (20) is obtained. Moreover, since $p(Y_i|\theta_k)$ and $p(\theta_k)$ are conjugate, $Q(\theta)$ has the same parametric form as $p(\theta)$.

A.3. $\frac{\partial \tilde{L}}{\partial Q(\phi)}$

Collect from $\tilde{L}(Q)$ the terms involving $Q(\phi)$ in $\tilde{L}_{Q(\phi)}$:

$$\begin{aligned} \tilde{L}_{Q(\phi)} &= E_{Q(\phi)}[\log p(\phi)] + E_{Q(\phi,\mathbf{Z})}[\log p(\mathbf{X}|\mathbf{Z}, \phi)] - E_{Q(\phi)}[\log Q(\phi)] + \lambda_\phi \int Q(\phi) \mathrm{d}\phi \\ &= \int Q(\phi)[\log p(\phi) + E_{Q(\mathbf{Z})}[\log p(\mathbf{X}|\mathbf{Z}, \phi)] - \log Q(\phi) + \lambda_\phi] \mathrm{d}\phi. \end{aligned}$$

Setting $\frac{\partial \tilde{L}}{\partial Q(\phi)} = 0$, we arrive at

$$Q(\phi) = \exp\{\log p(\phi) + E_{Q(\mathbf{Z})}[\log p(\mathbf{X}|\mathbf{Z}, \phi)]\} \Big/ C_\phi,$$

where λ_θ is related to the normalization constant C_θ as follows:

$$1 - \lambda_\phi = \log C_\phi = \log \int \exp\left\{\log p(\phi) + E_{Q(\mathbf{Z})}[\log p(\mathbf{X}|\mathbf{Z}, \phi)]\right\} \mathrm{d}\phi.$$

Since

$$\begin{aligned} \log p(\phi) &= \sum_{k \le l} \log p(\phi_{kl}), \\ E_{Q(\mathbf{Z})}[\log p(\mathbf{X}|\mathbf{Z}, \phi)] &= \sum_{i<j} E_{Q(Z_i, Z_j)}[\log p(X_{ij}|Z_i, Z_j, \phi)] \\ &= \sum_{i<j} \sum_{k,l} E_{Q(Z_i, Z_j)}[\delta(Z_i, k)\delta(Z_j, l)] \log p(X_{ij}|\phi_{kl}) \\ &= \sum_{i<j} \sum_{k,l} \tilde{\beta}_{ik}\tilde{\beta}_{jl} \log p(X_{ij}|\phi_{kl}) \\ &= \sum_k \sum_{i<j} \tilde{\beta}_{ik}\tilde{\beta}_{jk} \log p(X_{ij}|\phi_{kk}) + \sum_{k<l} \sum_{i \ne j} \tilde{\beta}_{ik}\tilde{\beta}_{jl} \log p(X_{ij}|\phi_{kl}), \end{aligned}$$

the exponent can be further expanded as

$$\begin{aligned} \log p(\phi) + E_{Q(\mathbf{Z})}[\log p(\mathbf{X}|\mathbf{Z}, \phi)] &= \sum_k \left[\log p(\phi_{kk}) + \sum_{i<j} \tilde{\beta}_{ik}\tilde{\beta}_{jk} \log p(X_{ij}|\phi_{kk}) \right] \\ &\quad + \sum_{k<l} \left\{ \log p(\phi_{kl}) + \sum_{i \ne j} \tilde{\beta}_{ik}\tilde{\beta}_{jl} \log p(X_{ij}|\phi_{kl}) \right\}. \end{aligned}$$

Thus, we have

$$Q(\phi|\tilde{\kappa}) = \prod_{k\le l} Q(\phi_{kl}|\tilde{\kappa}_{kl})$$

and

$$Q(\phi_{kk}|\tilde{\kappa}_{kk}) = \exp\left\{\log p(\phi_{kk}) + \sum_{i<j}\tilde{\beta}_{ik}\tilde{\beta}_{jk}\log p(X_{ij}|\phi_{kk})\right\}\Big/ C_{\phi_{kk}}, \quad k = 1, 2, \ldots, K,$$

$$Q(\phi_{kl}|\tilde{\kappa}_{kl}) = \exp\left\{\log p(\phi_{kl}) + \sum_{i\neq j}\tilde{\beta}_{ik}\tilde{\beta}_{jl}\log p(X_{ij}|\phi_{kl})\right\}\Big/ C_{\phi_{kl}}, \quad k, l = 1, 2, \ldots, K,\ k < l,$$

where

$$C_{\phi_{kk}} = \int \exp\left\{\log p(\phi_{kk}) + \sum_{i<j}\tilde{\beta}_{ik}\tilde{\beta}_{jk}\log p(X_{ij}|\phi_{kk})\right\} \mathrm{d}\phi_{kk}$$

$$C_{\phi_{kl}} = \int \exp\left\{\log p(\phi_{kl}) + \sum_{i\neq j}\tilde{\beta}_{ik}\tilde{\beta}_{jl}\log p(X_{ij}|\phi_{kl})\right\} \mathrm{d}\phi_{kl}.$$

Equations (21) and (22) are obtained. And similarly, $Q(\phi)$ has the same parametric form as $p(\phi)$.

A.4. $\frac{\partial \tilde{L}}{\partial Q(Z_i)}$

Collect from $\tilde{L}(Q)$ the terms involving $Q(Z_i)$ in $\tilde{L}_{Q(Z_i)}$:

$$\begin{aligned}\tilde{L}_{Q(Z_i)} &= E_{Q(\alpha,Z_i)}[\log p(Z_i|\alpha)] + E_{Q(\theta,Z_i)}[\log p(Y_i|Z_i,\theta)] \\ &\quad + \sum_{j\neq i} E_{Q(\phi,Z_i,Z_j)}[\log p(X_{ij}|Z_i, Z_j, \phi)] - E_{Q(Z_i)}[\log Q(Z_i)] + \lambda_i \sum_{Z_i} Q(Z_i) \\ &= \sum_{Z_i} Q(Z_i)\left\{E_{Q(\alpha)}[\log p(Z_i|\alpha)] + E_{Q(\theta)}[\log p(Y_i|Z_i,\theta)]\right. \\ &\quad \left. + \sum_{j\neq i} E_{Q(\phi,Z_j)}[\log p(X_{ij}|Z_i, Z_j, \phi)] - \log Q(Z_i) + \lambda_i\right\}.\end{aligned}$$

Setting $\frac{\partial \tilde{L}}{\partial Q(Z_i)} = 0$, we arrive at

$$Q(Z_i) = \exp\left\{E_{Q(\alpha)}[\log p(Z_i|\alpha)] + E_{Q(\theta)}[\log p(Y_i|Z_i,\theta)] + \sum_{j\neq i} E_{Q(\phi,Z_j)}[\log p(X_{ij}|Z_i, Z_j, \phi)]\right\}\Big/ C_i,$$

where λ_i is related to the normalization constant C_i as follows:

$$\log C_i = 1 - \lambda_i = \log \sum_{Z_i} \exp \left\{ E_{Q(\alpha)}[\log p(Z_i|\alpha)] + E_{Q(\theta)}[\log p(Y_i|Z_i, \theta)] + \sum_{j \neq i} E_{Q(\phi, Z_j)}[\log p(X_{ij}|Z_i, Z_j, \phi)] \right\}.$$

Since

$$\begin{aligned} E_{Q(\phi, Z_j)}[\log p(X_{ij}|Z_i, Z_j, \phi)] &= \sum_{k,l} E_{Q(\phi, Z_j)}[\delta(Z_i, k)\delta(Z_j, l) \log p(X_{ij}|\phi_{kl})] \\ &= \sum_k \delta(Z_i, k) \sum_l E_{Q(Z_j)}[\delta(Z_j, l)] E_{Q(\phi)}[\log p(X_{ij}|\phi_{kl})] \\ &= \sum_k \delta(Z_i, k) \sum_l \tilde{\beta}_{jl} E_{Q(\phi_{kl})}[\log p(X_{ij}|\phi_{kl})], \end{aligned}$$

the exponent can be further expanded as

$$\begin{aligned} &E_{Q(\alpha)}[\log p(Z_i|\alpha)] + E_{Q(\theta)}[\log p(Y_i|Z_i, \theta)] + \sum_{j \neq i} E_{Q(\phi, Z_j)}[\log p(X_{ij}|Z_i, Z_j, \phi)] \\ &= \sum_k \delta(Z_i, k) E_{Q(\alpha)}[\log \alpha_k] + \sum_k \delta(Z_i, k) E_{Q(\theta)}[\log p(Y_i|\theta_k)] \\ &\quad + \sum_{j \neq i} \sum_k \delta(Z_i, k) \sum_l \tilde{\beta}_{jl} E_{Q(\phi)}[\log p(X_{ij}|\phi_{kl})] \\ &= \sum_k \delta(Z_i, k) \left\{ E_{Q(\alpha)}[\log \alpha_k] + E_{Q(\theta_k)}[\log p(Y_i|\theta_k)] + \sum_{j \neq i} \sum_l \tilde{\beta}_{jl} E_{Q(\phi_{kl})}[\log p(X_{ij}|\phi_{kl})] \right\}. \end{aligned}$$

Thus, for any $i = 1, \ldots, N$, $Q(Z_i|\tilde{\beta}_i)$ is a multinomial distribution with parameters

$$\tilde{\beta}_{ik} = \exp \left\{ E_{Q(\alpha)}[\log \alpha_k] + E_{Q(\theta_k)}[\log p(Y_i|\theta_k)] + \sum_{j \neq i} \sum_l \tilde{\beta}_{jl} E_{Q(\phi_{kl})}[\log p(X_{ij}|\phi_{kl})] \right\} \Big/ C_i,$$

$k = 1, 2, \ldots, K$. Equation (23) is obtained.

B. PROOF FOR PROPOSITION 2

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

(1) Update $\tilde{\xi}^{(n)}$, $Q(\theta|\tilde{\gamma}^{(n)})$, $Q(\phi|\tilde{\kappa}^{(n)})$ according to Equations (19)–(22);
(2) Update $\tilde{\beta}^{(n)}$ according to Equation (23).

In order to prove $L(Q^{(n-1)}) \leq L(Q^{(n)})$, it suffices to show that none of the updating will decrease the value of L. We only prove this for updating $\tilde{\xi}^{(n)}$ in the following. The other cases can be shown similarly.

When updating $\tilde{\xi}^{(n)}$, we keep other variational distributions $Q(\theta|\tilde{\gamma}^{(n-1)})$, $Q(\phi|\tilde{\kappa}^{(n-1)})$, and parameter $\tilde{\beta}^{(n-1)}$ fixed. Thus, the aim is to show that

$$\begin{aligned}&L\big(Q(\alpha|\tilde{\xi}^{(n-1)}), Q(\theta|\tilde{\gamma}^{(n-1)}), Q(\phi|\tilde{\kappa}^{(n-1)}), Q(\mathbf{Z}|\tilde{\beta}^{(n-1)})\big)\\ &\quad\le L\big(Q(\alpha|\tilde{\xi}^{(n)}), Q(\theta|\tilde{\gamma}^{(n-1)}), Q(\phi|\tilde{\kappa}^{(n-1)}), Q(\mathbf{Z}|\tilde{\beta}^{(n-1)})\big).\end{aligned}$$

To this end, we consider

$$\begin{aligned}&L\big(\hat{Q}(\alpha), Q(\theta|\tilde{\gamma}^{(n-1)}), Q(\phi|\tilde{\kappa}^{(n-1)}), Q(\mathbf{Z}|\tilde{\beta}^{(n-1)})\big)\\ &\quad= E_{\hat{Q}(\alpha)}[\log p(\alpha)] + E_{\hat{Q}(\alpha)Q(\mathbf{Z}|\tilde{\beta}^{(n-1)})}[\log p(\mathbf{Z}|\alpha)] - E_{\hat{Q}(\alpha)}[\log \hat{Q}(\alpha)] + \mathcal{C}\\ &\quad= \int \hat{Q}(\alpha)\{\log p(\alpha) + E_{Q(\mathbf{Z}|\tilde{\beta}^{(n-1)})}[\log p(\mathbf{Z}|\alpha)] - \log \hat{Q}(\alpha)\}\mathrm{d}\alpha + \mathcal{C}\\ &\quad= -\mathrm{KL}(\hat{Q}(\alpha)||Q(\alpha|\tilde{\xi}^{(n)})) + \log C_{\alpha}^{(n-1)} + \mathcal{C},\end{aligned}$$

where $\mathcal{C}$ is a constant with respect to $\hat{Q}(\alpha)$ and

$$C_{\alpha}^{(n-1)} = \int \exp\left\{\log p(\alpha) + E_{Q(\mathbf{Z}|\tilde{\beta}^{(n-1)})}[\log p(\mathbf{Z}|\alpha)]\right\} \mathrm{d}\alpha.$$

Note that $\mathrm{KL}(\hat{Q}(\alpha)||Q(\alpha|\tilde{\xi}^{(n)}))$ is nonnegative and attains its minimum value zero if and only if $\hat{Q}(\alpha) = Q(\alpha|\tilde{\xi}^{(n)})$. Therefore, $L(\hat{Q}(\alpha), Q(\theta|\tilde{\gamma}^{(n-1)}), Q(\phi|\tilde{\kappa}^{(n-1)}), Q(\mathbf{Z}|\tilde{\beta}^{(n-1)}))$ will be maximized at $Q(\alpha|\tilde{\xi}^{(n)})$. That is, for any variational distribution $\hat{Q}(\alpha)$, we have

$$\begin{aligned}&L(\hat{Q}(\alpha), Q(\theta|\tilde{\gamma}^{(n-1)}), Q(\phi|\tilde{\kappa}^{(n-1)}), Q(\mathbf{Z}|\tilde{\beta}^{(n-1)}))\\ &\quad\le L(Q(\alpha|\tilde{\xi}^{(n)}), Q(\theta|\tilde{\gamma}^{(n-1)}), Q(\phi|\tilde{\kappa}^{(n-1)}), Q(\mathbf{Z}|\tilde{\beta}^{(n-1)})),\end{aligned}$$

which completes the proof.

C. PROOF FOR PROPOSITION 3

The quadratic time complexity of Algorithm 2 is caused by the updating of $Q(\phi)$ and $\tilde{\beta}$ (Equations (21)–(23)). Therefore, it suffices to show that the complexity of these updating steps can be reduced to $O(\mathrm{nnz}(\mathbf{X})K^2)$ under the assumptions of Proposition 3.

We start by defining the parametric forms of $p(X_{ij}|\phi_{kl})$ and its conjugate prior $p(\phi_{kl}|\kappa_{kl})$. Since $p(X_{ij}|\phi_{kl})$ is from the exponential family, it can be expressed as

$$p(X_{ij}|\phi_{kl}) = h(X_{ij})\exp\{\eta(\phi_{kl}) \cdot s(X_{ij}) - A(\phi_{kl})\}.$$

Here, $h(\cdot)$, $\eta(\cdot)$, $s(\cdot)$, and $A(\cdot)$ are the base measure, parameter function (a row vector), sufficient statistics (a column vector), and log-partition function of the exponential family, respectively. As a member of the exponential family, $p(X_{ij}|\phi_{kl})$ always has conjugate prior in theory, which is given by

$$p(\phi_{kl}|\kappa_{kl}) = h(\phi_{kl})\exp\{\kappa_{kl} \cdot s(\phi_{kl}) - A(\kappa_{kl})\}.$$

Here, $\kappa_{kl} = (\kappa_{kl}^{(1)}, \kappa_{kl}^{(2)})$ is the hyperparameter (a row vector) for ϕ_{kl}, while $s(\phi_{kl}) = (\eta(\phi_{kl}), -A(\phi_{kl}))^T$ is the corresponding sufficient statistics (a column vector).

Plugging these parametric representations of $p(X_{ij}|\phi_{kl})$ and $p(\phi_{kl}|\kappa_{kl})$ into Equations (21) and (22), we can see that the optimal variational distributions $Q(\phi_{kk})$ and $Q(\phi_{kl})$ now take the following forms:

$$\begin{aligned}Q(\phi_{kk}|\tilde{\kappa}_{kk}) &= h(\phi_{kk})\exp\{\tilde{\kappa}_{kk} \cdot s(\phi_{kk}) - A(\tilde{\kappa}_{kk})\},\\ Q(\phi_{kl}|\tilde{\kappa}_{kl}) &= h(\phi_{kl})\exp\{\tilde{\kappa}_{kl} \cdot s(\phi_{kl}) - A(\tilde{\kappa}_{kl})\},\end{aligned}$$

where

$$\tilde{\kappa}_{kk} = \left(\kappa_{kk}^{(1)} + \sum_{\substack{i,j=1 \\ i<j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jk} s(X_{ij})^T, \kappa_{kk}^{(2)} + \sum_{\substack{i,j=1 \\ i<j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jk} \right), \tag{34}$$

$$\tilde{\kappa}_{kl} = \left(\kappa_{kl}^{(1)} + \sum_{\substack{i,j=1 \\ i\neq j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jl} s(X_{ij})^T, \kappa_{kl}^{(2)} + \sum_{\substack{i,j=1 \\ i\neq j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jl} \right) \tag{35}$$

are the variational parameters. As a result, in order to update $Q(\phi_{kk})$ and $Q(\phi_{kl})$ in each iteration of Algorithm 2, we only need to update the variational parameters $\tilde{\kappa}_{kk}$ and $\tilde{\kappa}_{kl}$ according to Equations (34) and (35). By the assumption of Proposition 3, the sufficient statistic vectors $s(X_{ij})$ preserve the sparsity of $\mathbf{X}$. Therefore, the first components of $\tilde{\kappa}_{kk}$ and $\tilde{\kappa}_{kl}$ can be calculated in $O(\text{nnz}(\mathbf{X})K^2)$ time for all $k, l = \{1, \ldots, K\}$. Further note that

$$\sum_{\substack{i,j=1 \\ i<j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jl} = \sum_{i=1}^{N} \tilde{\beta}_{ik} \sum_{j=i+1}^{N} \tilde{\beta}_{jl},$$

$$\sum_{\substack{i,j=1 \\ i\neq j}}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{jl} = \left(\sum_{i=1}^{N} \tilde{\beta}_{ik} \right) \left(\sum_{j=1}^{N} \tilde{\beta}_{jl} \right) - \sum_{i=1}^{N} \tilde{\beta}_{ik}\tilde{\beta}_{il}.$$

Therefore, the second components of $\tilde{\kappa}_{kk}$ and $\tilde{\kappa}_{kl}$ can be calculated in $O(NK^2)$ time for all $k, l = \{1, \ldots, K\}$. Putting together, the overall complexity for updating $Q(\phi_{kk})$ and $Q(\phi_{kl})$ is reduced to $O(\text{nnz}(\mathbf{X})K^2)$.

The updating of $\tilde{\beta}$ enjoys a similar reduction in time complexity. Plugging the parametric form of $p(X_{ij}|\phi_{kl})$ into Equation (23), we obtain the following updating rule:

$$\tilde{\beta}_{ik} \propto \exp\left\{ E_{Q(\alpha)}[\log \alpha_k] + E_{Q(\theta_k)}[\log p(Y_i|\theta_k)] \right.$$

$$\left. + \sum_{\substack{j=1 \\ j\neq i}}^{N} s(X_{ij})^T \sum_{l=1}^{K} \tilde{\beta}_{jl} E_{Q(\phi_{kl})}[\eta(\phi_{kl})^T] - \sum_{l=1}^{K} E_{Q(\phi_{kl})}[A(\phi_{kl})] \sum_{\substack{j=1 \\ j\neq i}}^{N} \tilde{\beta}_{jl} \right\}.$$

Note that the time required to calculate the expectations $E_{Q(\alpha)}[\log \alpha_k]$, $E_{Q(\theta_k)}[\log p(Y_i|\theta_k)]$, $E_{Q(\phi_{kl})}[\eta(\phi_{kl})^T]$, and $E_{Q(\phi_{kl})}[A(\phi_{kl})]$ depends on the specific forms of the distributions but is constant in N and K. Furthermore, the sufficient statistic vectors $s(X_{ij})$ preserve the sparsity of $\mathbf{X}$ by assumption. Therefore, it is easy to verify that the complexity for updating $\tilde{\beta}$ is also reduced to $O(\text{nnz}(\mathbf{X})K^2)$, which completes the proof.

D. DERIVATION OF UPDATING RULES FOR UNWEIGHTED GRAPH

D.1. $Q(\theta|\tilde{\gamma})$

From Proposition 1,

$$Q(\theta_k) = \exp\left\{\log p(\theta_k) + \sum_i \tilde{\beta}_{ik} \log p(Y_i|\theta_k)\right\} \Big/ C_{\theta_k}, \quad k = 1, 2, \ldots, K,$$

where

$$\begin{aligned}\log p(\theta_k) + \sum_i \tilde{\beta}_{ik} \log p(Y_i|\theta_k) &= \sum_t p(\theta_{kt}) + \sum_i \sum_t \tilde{\beta}_{ik} \log p(Y_{it}|\theta_{kt}) \\ &= \sum_t \left[p(\theta_{kt}) + \sum_i \tilde{\beta}_{ik} \log p(Y_{it}|\theta_{kt}) \right].\end{aligned}$$

Thus,

$$Q(\theta_k|\tilde{\gamma}_k) = \prod_t Q(\theta_{kt}|\tilde{\gamma}_{kt}),$$

and

$$Q(\theta_{kt}|\tilde{\gamma}_{kt}) = \exp\left\{\log p(\theta_{kt}) + \sum_i \tilde{\beta}_{ik} \log p(Y_{it}|\theta_{kt})\right\} \Big/ C_{\theta_{kt}},$$

where $C_{\theta_{kt}} = \int \exp\{\log p(\theta_{kt}) + \sum_i \tilde{\beta}_{ik} \log p(Y_{it}|\theta_{kt})\} \mathrm{d}\theta_{kt}$.

Expanding the exponent of $Q(\theta_{kt})$, we have

$$\begin{aligned}&\log p(\theta_{kt}) + \sum_i \tilde{\beta}_{ik} \log p(Y_{it}|\theta_{kt}) \\ &= \log\left\{\frac{\Gamma\left(\sum_m \gamma_{tm}\right)}{\prod_m \Gamma\left(\gamma_{tm}\right)}\right\} + \sum_m \left(\gamma_{tm} - 1\right) \log \theta_{ktm} + \sum_i \tilde{\beta}_{ik} \sum_m \delta\left(Y_{it}, a_{tm}\right) \log\left(\theta_{ktm}\right) \\ &= \log\left\{\frac{\Gamma\left(\sum_m \gamma_{tm}\right)}{\prod_m \Gamma\left(\gamma_{tm}\right)}\right\} + \sum_m \left\{\left[\gamma_{tm} + \sum_i \tilde{\beta}_{ik} \delta\left(Y_{it}, a_{tm}\right) - 1\right] \log \theta_{ktm}\right\}.\end{aligned}$$

Therefore, for any $t = 1, 2, \ldots, T$, $k = 1, 2, \ldots, K$, $Q(\theta_{kt}|\tilde{\gamma}_{kt})$ is a Dirichlet distribution with parameters

$$\tilde{\gamma}_{ktm} = \gamma_{tm} + \sum_i \tilde{\beta}_{ik} \delta\left(Y_{it}, a_{tm}\right), \quad m = 1, 2, \ldots, M^t.$$

D.2. $Q(\phi|\tilde{\mu}, \tilde{\nu})$

From Proposition 1, for any $k, l = 1, 2, \ldots, K$ and $k < l$,

$$Q(\phi_{kl}|\tilde{\mu}_{kl}, \tilde{\nu}_{kl}) = \exp\left\{\log p(\phi_{kl}) + \sum_{i \neq j} \tilde{\beta}_{ik} \tilde{\beta}_{jl} \log p(X_{ij}|\phi_{kl})\right\} \Big/ C_{\phi_{kl}}.$$

The exponent can be expanded as

$$\begin{aligned}
&\log p(\phi_{kl}) + \sum_{i\neq j} \tilde{\beta}_{ik}\tilde{\beta}_{jl} \log p(X_{ij}|\phi_{kl}) \\
&= \log \frac{\Gamma(\mu+\nu)}{\Gamma(\mu)\Gamma(\nu)} + (\mu-1)\log\phi_{kl} + (\nu-1)\log(1-\phi_{kl}) \\
&\quad + \sum_{i\neq j} \tilde{\beta}_{ik}\tilde{\beta}_{jl}\big[(1-X_{ij})\log(1-\phi_{kl}) + X_{ij}\log\phi_{kl}\big] \\
&= \log \frac{\Gamma(\mu+\nu)}{\Gamma(\mu)\Gamma(\nu)} + \left(\mu + \sum_{i\neq j} \tilde{\beta}_{ik}\tilde{\beta}_{jl}X_{ij} - 1\right)\log\phi_{kl} \\
&\quad + \left[\nu + \sum_{i\neq j} \tilde{\beta}_{ik}\tilde{\beta}_{jl}(1-X_{ij}) - 1\right]\log(1-\phi_{kl}),
\end{aligned}$$

which implies that $Q(\phi_{kl})$ is a Beta distribution with parameters $\tilde{\mu}_{kl} = \mu + \sum_{i\neq j} \tilde{\beta}_{ik}\tilde{\beta}_{jl}X_{ij}$ and $\tilde{\nu}_{kl} = \nu + \sum_{i\neq j} \tilde{\beta}_{ik}\tilde{\beta}_{jl}(1-X_{ij})$, for $k, l = 1, 2, \ldots, K,\ k < l$.

Similarly, $Q(\phi_{kk})$ is a Beta distribution with parameters $\tilde{\mu}_{kk} = \mu + \sum_{i<j} \tilde{\beta}_{ik}\tilde{\beta}_{jk}X_{ij}$ and $\tilde{\nu}_{kk} = \nu + \sum_{i<j} \tilde{\beta}_{ik}\tilde{\beta}_{jk}(1-X_{ij})$, for $k = 1, 2, \ldots, K$.

D.3. $Q(Z|\tilde{\beta})$

From Proposition 1, for any $i = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, K$,

$$\tilde{\beta}_{ik} = \exp\left\{E_{Q(\alpha)}[\log\alpha_k] + E_{Q(\theta_k)}[\log p(Y_i|\theta_k)] + \sum_{j\neq i}\sum_{l} \tilde{\beta}_{jl}E_{Q(\phi_{kl})}[\log p(X_{ij}|\phi_{kl})]\right\}\Big/ C_i.$$

For the items in exponents, we have

$$\begin{aligned}
&E_{Q(\alpha|\tilde{\xi})}[\log\alpha_k] \\
&= \int Q(\alpha|\tilde{\xi})\log\alpha_k \mathrm{d}\alpha = \frac{\partial}{\partial\tilde{\xi}_k}\left[\sum_l \log\Gamma(\tilde{\xi}_l) - \log\Gamma\left(\sum_r \tilde{\xi}_r\right)\right] = \psi(\tilde{\xi}_k) - \psi\left(\sum_r \tilde{\xi}_r\right), \\
&E_{Q(\theta_k)}[\log p(Y_i|\theta_k)] \\
&= \sum_t\sum_m \delta(Y_{it}, a_{tm})E_{Q(\theta_{kt})}[\log\theta_{ktm}] = \sum_t\sum_m \delta(Y_{it}, a_{tm})\left[\psi(\tilde{\gamma}_{ktm}) - \psi\left(\sum_n \tilde{\gamma}_{ktn}\right)\right], \\
&E_{Q(\phi_{kl})}[\log p(X_{ij}|\phi_{kl})] \\
&= (1-X_{ij})E_{Q(\phi_{kl})}[\log(1-\phi_{kl})] + X_{ij}E_{Q(\phi_{kl})}[\log\phi_{kl}] \\
&= (1-X_{ij})[\psi(\tilde{\nu}_{kl}) - \psi(\tilde{\mu}_{kl}+\tilde{\nu}_{kl})] + X_{ij}[\psi(\tilde{\mu}_{kl}) - \psi(\tilde{\mu}_{kl}+\tilde{\nu}_{kl})] \\
&= X_{ij}\psi(\tilde{\mu}_{kl}) + (1-X_{ij})\psi(\tilde{\nu}_{kl}) - \psi(\tilde{\mu}_{kl}+\tilde{\nu}_{kl}).
\end{aligned}$$

Thus,

$$\tilde{\beta}_{ik} \propto \exp\left\{\psi(\tilde{\xi}_k) - \psi\left(\sum_l \tilde{\xi}_l\right) + \sum_t \sum_m \delta(Y_{it}, a_{tm})\left[\psi(\tilde{\gamma}_{ktm}) - \psi\left(\sum_n \tilde{\gamma}_{ktn}\right)\right] + \sum_{j\neq i}\sum_l \tilde{\beta}_{jl}[X_{ij}\psi(\tilde{\mu}_{kl}) + (1-X_{ij})\psi(\tilde{\nu}_{kl}) - \psi(\tilde{\mu}_{kl}+\tilde{\nu}_{kl})]\right\} \Big/ C_i.$$

E. DERIVATION OF UPDATING RULES FOR WEIGHTED GRAPH

E.1. $Q(\phi|\tilde{\mu},\tilde{\nu})$

From Proposition 1, for any $k, l = 1, 2, \ldots, K$ and $k < l$,

$$Q(\phi_{kl}|\tilde{\mu}_{kl},\tilde{\nu}_{kl}) = \exp\left\{\log p(\phi_{kl}) + \sum_{i\neq j}\tilde{\beta}_{ik}\tilde{\beta}_{jl}\log p(X_{ij}|\phi_{kl})\right\} \Big/ C_{\phi_{kl}}.$$

The exponent can be expanded as

$$\begin{aligned}
&\log p(\phi_{kl}) + \sum_{i\neq j}\tilde{\beta}_{ik}\tilde{\beta}_{jl}\log p(X_{ij}|\phi_{kl}) \\
&= \mu\log\nu - \log\Gamma(\mu) + (\mu-1)\log\phi_{kl} - \nu\phi_{kl} + \sum_{i\neq j}\tilde{\beta}_{ik}\tilde{\beta}_{jl}(X_{ij}\log\phi_{kl} - \phi_{kl}) \\
&= \mu\log\nu - \log\Gamma(\mu) + \left(\mu + \sum_{i\neq j}\tilde{\beta}_{ik}\tilde{\beta}_{jl}X_{ij} - 1\right)\log\phi_{kl} - \left(\nu + \sum_{i\neq j}\tilde{\beta}_{ik}\tilde{\beta}_{jl}\right)\phi_{kl},
\end{aligned}$$

which implies that $Q(\phi_{kl})$ is a Gamma distribution with parameters $\tilde{\mu}_{kl} = \mu + \sum_{i\neq j}\tilde{\beta}_{ik}\tilde{\beta}_{jl}X_{ij}$ and $\tilde{\nu}_{kl} = \nu + \sum_{i\neq j}\tilde{\beta}_{ik}\tilde{\beta}_{jl}$, for $k, l = 1, \ldots, K$, $k < l$.

Similarly, $Q(\phi_{kk})$ is a Gamma distribution with parameters $\tilde{\mu}_{kk} = \mu + \sum_{i<j}\tilde{\beta}_{ik}\tilde{\beta}_{jk}X_{ij}$ and $\tilde{\nu}_{kk} = \nu + \sum_{i<j}\tilde{\beta}_{ik}\tilde{\beta}_{jk}$, for $k = 1, \ldots, K$.

E.2. $Q(Z|\tilde{\beta})$

From Proposition 1, for any $i = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, K$,

$$\tilde{\beta}_{ik} = \exp\left\{E_{Q(\alpha)}[\log\alpha_k] + E_{Q(\theta_k)}[\log p(Y_i|\theta_k)] + \sum_{j\neq i}\sum_l \tilde{\beta}_{jl}E_{Q(\phi_{kl})}[\log p(X_{ij}|\phi_{kl})]\right\} \Big/ C_i\,.$$

We only need to figure out the expectation of $\log p(X_{ij}|\phi_{kl})$ with respect to $Q(\phi_{kl})$ now. Notice that the expectation of Gamma distribution $f(x|a,b) = \frac{b^a}{\Gamma(a)}x^{a-1}\exp\{-bx\}$ is $\frac{a}{b}$. If we write Gamma distribution in the form of exponential family

$$f(x|a,b) = \exp\{[(a-1)\log x - bx] + a\log b - \log\Gamma(a)\},$$

then

$$E[\log x] = \frac{\partial}{\partial a}[\log\Gamma(a) - a\log b] = \psi(a) - \log b.$$

Thus,

$$\begin{aligned}
E_{Q(\phi_{kl})}[\log p(X_{ij}|\phi_{kl})] &= -E_{Q(\phi_{kl})}[\phi_{kl}] + X_{ij}E_{Q(\phi_{kl})}[\log\phi_{kl}] - \log(X_{ij}!) \\
&= -\frac{\tilde{\mu}_{kl}}{\tilde{\nu}_{kl}} + X_{ij}[\psi(\tilde{\mu}_{kl}) - \log\tilde{\nu}_{kl}] - \log(X_{ij}!).
\end{aligned}$$

Consequently,

$$\tilde{\beta}_{ik} = \exp\left\{\psi(\tilde{\xi}_k) - \psi\left(\sum_l \tilde{\xi}_l\right) + \sum_t \sum_m \delta(Y_{it}, a_{tm})\left[\psi(\tilde{\gamma}_{ktm}) - \psi\left(\sum_n \tilde{\gamma}_{ktn}\right)\right] + \sum_{j\neq i}\sum_l \tilde{\beta}_{jl}\left[-\frac{\tilde{\mu}_{kl}}{\tilde{\nu}_{kl}} + X_{ij}\left[\psi(\tilde{\mu}_{kl}) - \log\tilde{\nu}_{kl}\right] - \log(X_{ij}!)\right]\right\} \Big/ C_i.$$

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments that have helped improve the quality of the article significantly.

REFERENCES

Leman Akoglu, Hanghang Tong, Brendan Meeder, and Christos Faloutsos. 2012. PICS: Parameter-free identification of cohesive subgroups in large attributed graphs. In *SDM*. 439–450.

Matthew J. Beal. 2003. *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. Dissertation. Gatsby Computational Neuroscience Unit, University College London.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent dirichlet allocation. In *NIPS*. 601–608.

GeorgeC Casella and Roger Berger. 2001. *Statistical Inference*. Duxbury Resource Center.

Hong Cheng, Yang Zhou, and Jeffrey Xu Yu. 2011. Clustering large attributed graphs: A balance between structural and attribute similarities. *TKDD* 5, 2 (2011), 12.

David Cohn and Thomas Hofmann. 2001. The missing link—A probabilistic model of document content and hypertext connectivity. In *NIPS*.

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. Wiley-Interscience.

M. H. DeGroot. 1986. *Proability and Statistics* (2nd ed.). Addison-Wesley.

Elena Erosheva, Steve Fienberg, and John Lafferty. 2004. Mixed membership models of scientific publications. In *PNAS*.

A. Gelman. 2006. Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis* 1, 3 (2006), 515–534.

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. 2003. *Bayesian Data Analysis* (2nd ed.). CRC Press.

Xiaofeng He, Chris H. Q. Ding, Hongyuan Zha, and Horst D. Simon. 2001. Automatic topic identification using webpage clustering. In *ICDM*. 195–202.

Keith Henderson, Tina Eliassi-Rad, Spiros Papadimitriou, and Christos Faloutsos. 2010. HCDF: A hybrid community discovery framework. In *SDM*. 754–765.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *SIGIR*. 50–57.

Michael I. Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence K. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning* 37, 2 (1999), 183–233.

George Karypis and Vipin Kumar. 1998. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing* 48 (1998), 96–129.

Ramesh Nallapati, Amr Ahmed, Eric P. Xing, and William W. Cohen. 2008. Joint latent topic models for text and citations. In *KDD*. 542–550.

Jennifer Neville, Micah Adler, and David Jensen. 2003. Clustering relational data using attribute and link information. In *Text Mining and Link Analysis Workshop, IJCAI*. 689–698.

M. E. J. Newman. 2004a. Analysis of weighted networks. *Physical Review E* 70 (2004).

M. E. J. Newman. 2004b. Fast algorithm for detecting community structure in networks. *Physical Review E* 69 (2004), 066133.

M. E. J. Newman and M. Girvan. 2004. Finding and evaluating community structure in networks. *Physical Review E* 69 (2004), 066113.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Karsten Steinhaeuser and Nitesh V. Chawla. 2008. Community detection in a large real-world social network. In *Social Computing, Behavioral Modeling, and Prediction*. 168–175.

Yizhou Sun, Charu C. Aggarwal, and Jiawei Han. 2012. Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. *PVLDB* 5, 5 (2012), 394–405.

Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *ICDM*. 613–622.

Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. 2012. A model-based approach to attributed graph clustering. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD'12)*. ACM, New York, NY, 505–516. DOI:http://dx.doi.org/10.1145/2213836.2213894

Tianbao Yang, Rong Jin, Yun Chi, and Shenghuo Zhu. 2009a. A bayesian framework for community detection integrating content and link. In *UAI*. 615–622.

Tianbao Yang, Rong Jin, Yun Chi, and Shenghuo Zhu. 2009b. Combining link and content for community detection: a discriminative approach. In *KDD*. 927–936.

Hugo Zanghi, Stevenn Volant, and Christophe Ambroise. 2010. Clustering based on random graph model embedding vertex features. *Pattern Recognition Letters* 31, 9 (2010), 830–836.

Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2009. Graph clustering based on structural/attribute similarities. *PVLDB* 2, 1 (2009), 718–729.

Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. 2010. Clustering large attributed graphs: An efficient incremental approach. In *ICDM*. 689–698.

Received June 2013; revised December 2013; accepted February 2014