Electronic Preprint for Journal of Information Processing Vol.20 No.4

Invited Paper

Clustering Large Attributed Graph

Hong Cheng[1,a) Jeffrey Xu Yu[1,b)

Received: May 12, 2012, Accepted: July 19, 2012

Abstract: Graph clustering is a long-standing problem in data mining and machine learning. Traditional graph clustering aims to partition a graph into several densely connected components. However, with the proliferation of rich attribute information available for objects in real-world graphs, vertices in graphs are often associated with a number of attributes that describe the properties of the vertices. This gives rise to a new type of graphs, namely *attributed graphs*. Thus, how to leverage structural and attribute information becomes a new challenge for *attributed graph clustering*. In this paper, we introduce the state-of-the-art studies on clustering large attributed graphs. These methods propose different approaches to leverage both structural and attribute information. The resulting clusters will have both cohesive intra-cluster structures and homogeneous attribute values.

Keywords: attributed graph clustering, random walk, model-based clustering, Bayesian method, graph summarization

1. Introduction

Graph as an expressive data structure is popularly used to model structural relationship between objects in many application domains such as web, social networks, sensor networks and telecommunication, etc. Graph clustering is an interesting and challenging research problem which has received much attention recently [7], [9], [10], [14]. Traditional graph clustering methods aim to partition a graph into several densely connected components. Such methods build clusters based on normalized cuts [10], modularity [7], structural density [14], or stochastic flows [9]. Typical applications of graph clustering include community detection in social networks, identification of functional related protein modules in large protein-protein interaction networks, etc.

In recent years, with the proliferation of rich information available for real-world objects, vertices in graphs are often associated with a number of attributes that describe the properties of the vertices. This gives rise to a new type of graphs, namely *attributed graphs*, and hence the demand of a new clustering task, *attributed graph clustering*. Attributed graph clustering can find many real applications. For example, in a social network, vertex attributes describe roles of a person while the topological structure represents relationships among a group of people. Directly applying traditional graph clustering methods on attributed graphs will completely ignore the rich attribute information in clustering. A more reasonable clustering result should have a *cohesive* intra-cluster structure with *homogeneous* vertex properties, by balancing the structural and attribute information. Let us look at the following example.

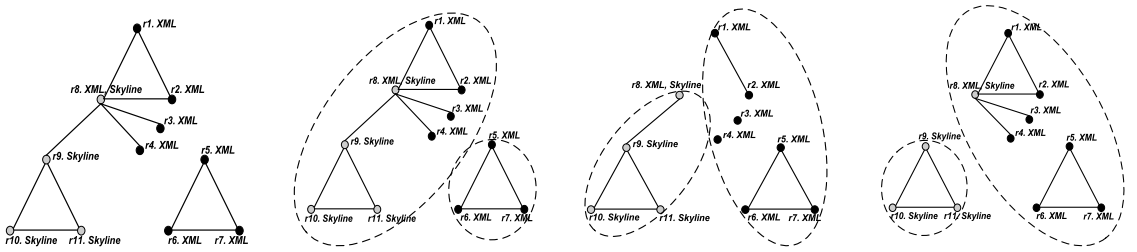
Figure 1 (a) shows a coauthor graph where a vertex represents an author and an edge represents the coauthor relationship between two authors. In addition, there are an author ID and research topic(s) associated with each author. The research topic is considered as an attribute to describe the vertex property. As we can see, authors r_1–r_7 work on XML, authors r_9–r_{11} work on $Skyline$ and r_8 works on both. Given a cluster number $k = 2$, we could partition the graph into 2 clusters in several possible ways depending on the clustering criteria:

- **Structure-based Clustering**. Figure 1 (b) shows a clustering result based on vertex connectivity, i.e., coauthor relationship. Authors within clusters are closely connected; however, they can have quite different topics, e.g., half work on XML and the other half work on $Skyline$ in one of the clusters.
- **Attribute-based Clustering**. Figure 1 (c) shows another clustering result based on attribute similarity, i.e., topics. Authors within clusters work on the same topics; however, the coauthor relationship may be lost due to the partitioning so that authors are quite isolated in one of the clusters.
- **Structural and Attribute Clustering**. Figure 1 (d) shows the clustering result based on both structure and attribute information. This clustering result balances the structural and attribute similarities: authors within one cluster are closely connected; meanwhile, they are homogeneous on research topics.

In this paper, we will introduce the state-of-the-art attributed graph clustering methods, including SA-Cluster [17] and Inc-Cluster [18] which use random walk to unify the structure and attribute information, and BAGC [15] which takes a model-based approach to cluster attributed graphs. The goal of these methods is to partition an attributed graph into k clusters with cohesive intra-cluster structures and homogeneous attribute values.

The rest of this paper is organized as follows. Section 2 introduces the preliminary concepts and formulates the attributed

[1] Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong
[a)] hcheng@se.cuhk.edu.hk
[b)] yu@se.cuhk.edu.hk



(a) Co-author Graph (b) Structure-based Cluster (c) Attribute-based Cluster (d) Structural/Attribute Cluster

Fig. 1 A coauthor network example with an attribute "topic."

graph clustering problem. Section 3 describes SA-Cluster [17] and its improved version Inc-Cluster [18]. Section 4 introduces BAGC [15]. Section 5 briefly describes other related works on attributed graph clustering. Finally, Section 6 concludes the paper.

2. Preliminary Concepts

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

Given an attributed graph G and the number of clusters K, the attributed graph clustering problem is to partition the vertex set V of G into K disjoint subsets $V_1, V_2, \ldots, V_K$, where $V = \bigcup_{i=1}^{K} V_i$ and $V_i \cap V_j = \emptyset$ for any $i \ne j$, such that: (1) vertices within one cluster are densely connected, while vertices in different clusters are loosely connected; and (2) vertices within one cluster have similar attribute values, while vertices in different clusters can have quite different attribute values.

3. SA-Cluster: A Random Walk Based Approach

The SA-Cluster method proposes to use graph augmentation to express the attribute similarity by edge connectivity. In this way it naturally combines the structural closeness and attribute similarity through the random walk distance measure. The *attribute augmented graph* is defined as follows.

Definition 1 (Attribute Augmented Graph) Given an attributed graph $G = (V, E, \Lambda, F)$ with a set of attributes $\Lambda = \{a^1, a^2, \ldots, a^T\}$. The domain of attribute a^i is $dom(a^i) = \{a^i_1, \ldots, a^i_{n_i}\}$ with a size of $|dom(a^i)| = n_i$. An attribute augmented graph is denoted as $G_a = (V \cup V_a, E \cup E_a)$ where $V_a = \{v_{ij}\}_{i=1,j=1}^{T,\ n_i}$ is the set of attribute vertices and $E_a \subseteq V \times V_a$ is the set of attribute edges. An attribute vertex $v_{ij} \in V_a$ represents that attribute a^i takes the value a^i_j. An attribute edge $(v_i, v_{jk}) \in E_a$ iff $f_j(v_i) = a^j_k$, i.e., vertex v_i takes the value of a^j_k on attribute a^j. Accordingly, a vertex $v \in V$ is called a structure vertex and an edge $(v_i, v_j) \in E$ is called a structure edge.

Figure 2 is an attribute augmented graph on the author-topic example. Two attribute vertices v_{11} and v_{12} representing the topics "XML" and "Skyline" are added. Authors with corresponding

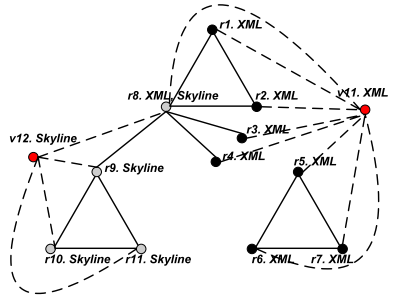


Fig. 2 Attribute augmented graph with topics.

topics are connected to the two vertices respectively in dashed lines. With the attribute edges, authors who are originally isolated become much closer if they share a common topic, e.g., r_1 and r_5.

3.1 A Unified Random Walk Distance

SA-Cluster uses the neighborhood random walk model on the attribute augmented graph G_a to compute a unified distance between vertices in V. The random walk distance between two vertices $v_i, v_j \in V$ is based on the paths consisting of both structure and attribute edges. Thus it effectively combines the structural proximity and attribute similarity of two vertices into one unified measure. The transition probability matrix P_A on G_a is defined as follows.

A structure edge $(v_i, v_j) \in E$ is of a different type from an attribute edge $(v_i, v_{jk}) \in E_a$. The T attributes may also have different importance. Therefore, they may have different degree of contributions in random walk distance. Without loss of generality, we assume that a structure edge has a weight of ω_0, attribute edges corresponding to a^1, a^2, ..., a^T have an edge weight of $\omega_1, \omega_2, \ldots, \omega_T$, respectively. Therefore, the transition probability from vertex v_i to vertex v_j through a structure edge is

$$p_{v_i,v_j} = \begin{cases} \dfrac{\omega_0}{|N(v_i)|\omega_0 + \sum_{i=1}^{T} \omega_i}, & \text{if } (v_i, v_j) \in E \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where $N(v_i)$ represents the set of structure vertices connected to v_i. Similarly, the transition probability from v_i to v_{jk} through an attribute edge is

$$p_{v_i,v_{jk}} = \begin{cases} \dfrac{\omega_j}{|N(v_i)|\omega_0 + \sum_{i=1}^{T} \omega_i}, & \text{if } (v_i, v_{jk}) \in E_a \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

The transition probability from v_{ik} to v_j through an attribute edge is

$$p_{v_{ik},v_j} = \begin{cases} \dfrac{1}{|N(v_{ik})|}, & \text{if } (v_{ik}, v_j) \in E_a \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The transition probability between two attribute vertices v_{ip} and v_{jq} is 0 as there is no edge between attribute vertices.

$$p_{v_{ip},v_{jq}} = 0, \forall v_{ip}, v_{jq} \in V_a \quad (4)$$

The transition probability matrix P_A is a $|V \cup V_a| \times |V \cup V_a|$ matrix, where the first $|V|$ rows (columns) correspond to the structure vertices and the rest $|V_a|$ rows (columns) correspond to the attribute vertices. For the ease of presentation, P_A is represented as

$$P_A = \begin{bmatrix} P_{V_1} & A_1 \\ B_1 & O \end{bmatrix} \quad (5)$$

where P_{V_1} is a $|V| \times |V|$ matrix representing the transition probabilities defined by Eq. (1); A_1 is a $|V| \times |V_a|$ matrix representing the transition probabilities defined by Eq. (2); B_1 is a $|V_a| \times |V|$ matrix representing the transition probabilities defined by Eq. (3); and O is a $|V_a| \times |V_a|$ zero matrix.

Definition 2 (Random Walk Matrix) Let P_A be the transition probability matrix of an attribute augmented graph G_a. Given L as the length that a random walk can go, $c \in (0, 1)$ as the random walk restart probability, the unified neighborhood random walk distance matrix R_A is

$$R_A = \sum_{l=1}^{L} c(1-c)^l P_A^l \quad (6)$$

3.2 Clustering Algorithm

With the random walk distance measure, SA-Cluster adopts the *K-Medoids* clustering method [5]: it selects the most centrally located point in a cluster as a centroid, and assigns the rest of points to their closest centroids. In each iteration, the edge weights $\{\omega_1, \ldots, \omega_T\}$ are adjusted to reflect the clustering tendencies of attributes. This process is repeated until convergence.

3.2.1 Cluster Centroid Initialization

Good initial centroids are essential for the success of partitioning clustering algorithms such as K-Means and K-Medoids. Instead of selecting initial centroids randomly, SA-Cluster identifies good initial centroids from the density point of view [3]. If the L-step neighborhood of a vertex v_i is dense, it means many vertices are reachable from v_i within L steps. Then v_i has a high probability of being in a dense cluster. The *influence function* is defined as follows.

Definition 3 (Influence Function) Let σ be a user-specified parameter. The influence function of one vertex v_i on another vertex v_j is defined as

$$f_B^{v_j}(v_i) = 1 - e^{-\frac{R_A(v_i,v_j)^2}{2\sigma^2}} \quad (7)$$

The influence function $f_B^{v_j}(v_i) \in [0, 1]$ measures the extent one vertex influences another one. The influence of v_i on v_j is a function of the random walk distance $R_A(v_i, v_j)$. The larger the random walk distance from v_i to v_j, the more influence v_i has on v_j.

Definition 4 (Density Function) The density function of one vertex v_i is the sum of the influence function of v_i on all vertices in V

$$f_B^D(v_i) = \sum_{v_j \in V} f_B^{v_j}(v_i) = \sum_{v_j \in V} \left(1 - e^{-\frac{R_A(v_i,v_j)^2}{2\sigma^2}}\right) \quad (8)$$

Then the vertices are sorted in the descending order of their density values and the densest K vertices are selected as the initial centroids $\{c_1^0, \ldots, c_K^0\}$.

3.2.2 Clustering Process

With K centroids in the t^{th} iteration, we assign each vertex $v_i \in V$ to its closest centroid $c^* \in \{c_1^t, \ldots, c_K^t\}$, i.e., a centroid c^* with the largest random walk distance from v_i:

$$c^* = \arg\max_{c_j^t} R_A(v_i, c_j^t)$$

When all vertices are assigned to some cluster, the centroid will be updated with the most centrally located vertex in each cluster. We denote the "mean point" of a cluster V_i as $\overline{v_i}$. The random walk distance from $\overline{v_i}$ to a vertex v_j is computed by

$$R_A(\overline{v_i}, v_j) = \frac{1}{|V_i|} \sum_{v_k \in V_i} R_A(v_k, v_j). \quad (9)$$

Accordingly, we can obtain a random walk distance vector $[R_A(\overline{v_i}, v_1), \ldots, R_A(\overline{v_i}, v_N)]$ from the mean point $\overline{v_i}$ to all vertices in V. We use $R_A(\overline{v_i})$ to denote the above random walk distance vector from the mean point of cluster V_i. Then we can find the new centroid c_i^{t+1} in cluster V_i as

$$c_i^{t+1} = \arg\min_{v_j \in V_i} \|R_A(v_j) - R_A(\overline{v_i})\| \quad (10)$$

Thus we find the new centroid c_i^{t+1} in the $(t+1)^{th}$ iteration whose random walk distance vector is the closest to the cluster average. The clustering process iterates until the clustering objective function converges.

3.2.3 Attribute Weight Self-Adjustment

As different attributes may have different importance in clustering, SA-Cluster designs an adaptive weight adjustment method. First, the structure edge weight is fixed at $\omega_0 = 1.0$. The attribute weights $\{\omega_1, \ldots, \omega_T\}$ are iteratively adjusted relative to ω_0. Let $W^t = \{\omega_1^t, \ldots, \omega_T^t\}$ be the attribute weights in the t^{th} iteration. It initializes $\omega_1^0 = \omega_2^0 = \ldots = \omega_T^0 = 1.0$, and iteratively adjusts ω_i^t with an increment $\triangle\omega_i^t$, which denotes the weight update of attribute a^i between the t^{th} iteration and the $(t+1)^{th}$. The weight of attribute a^i in the $(t+1)^{th}$ iteration is computed as

$$\omega_i^{t+1} = \frac{1}{2}(\omega_i^t + \triangle\omega_i^t) \quad (11)$$

To accurately determine the extent of weight increment $\triangle\omega_i$, SA-Cluster designs a majority vote mechanism: if a large portion of vertices within clusters share the same value of a certain attribute a^i, it means that a^i has a good clustering tendency. Then the weight ω_i of a^i is increased; on the other hand, if vertices within clusters have a very random distribution on values of a certain attribute a^i, then a^i is not a good clustering attribute. The weight ω_i should be decreased. We define a *vote* measure which determines whether two vertices share an attribute value.

$$vote_i(v_p, v_q) = \begin{cases} 1, & \text{if } v_p, v_q \textit{ share the } a^i \textit{ value} \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

Then $\triangle\omega_i^t$ is estimated by counting the number of vertices within clusters which share attribute values with the centroids on a^i. The larger number of vertices which share attribute values, the larger $\triangle\omega_i^t$ is.

$$\triangle\omega_i^t = \frac{\sum_{j=1}^{K}\sum_{v\in V_j} vote_i(c_j, v)}{\frac{1}{T}\sum_{p=1}^{T}\sum_{j=1}^{K}\sum_{v\in V_j} vote_p(c_j, v)} \quad (13)$$

The denominator in Eq. (13) ensures that the constraint $\sum_{i=1}^{T}\omega_i^{t+1} = T$ is still satisfied after weight adjustment. Combining Eq. (11) and Eq. (13) gives the weight adjustment formula. Reference [17] shows that *the attribute weights are adjusted towards the direction of increasing the clustering objective function value till convergence*.

3.3 Inc-Cluster: Speedup Random Walk Computation

As the edge weights are iteratively adjusted to balance the importance between structural and attribute similarities in SA-Cluster, matrix multiplication is repeated in each iteration of the clustering process to recalculate the random walk distances which are affected by the edge weight update. In order to improve the efficiency and scalability of SA-Cluster, Zhou et al. proposed Inc-Cluster [18] to incrementally update the random walk distances given the edge weight increments. The main idea is to compute the full random walk distance matrix R_A only once at the beginning of the clustering process. Then in each following iteration of clustering, given the attribute weight increments $\{\triangle\omega_1, \ldots, \triangle\omega_T\}$, we use Inc-Cluster to efficiently calculate the increment matrix $\triangle R_A$, and then get the updated random walk distance matrix $R_A + \triangle R_A$. In this process, we only calculate the non-zero elements in $\triangle R_A$, i.e., those elements which are affected by the edge weight changes, but can ignore the unaffected parts of the original matrix. If the number of affected matrix elements is small, this incremental approach will be much more efficient than calculating the full matrix R_A from scratch in each iteration. By analyzing how the transition probabilities are affected by the weight increments, the random walk distance matrix is divided into four submatrices for incremental update. Inc-Cluster can speedup the clustering process significantly, and achieves the same clustering result.

4. BAGC: A Bayesian Model Based Approach

Xu et al. proposed BAGC [15], a model-based approach for attributed graph clustering. They develop a Bayesian probabilistic model for attributed graphs, and then formulate the clustering problem as a probabilistic inference problem. This work takes a variational approach and designs an efficient approximate algorithm to solve the inference problem.

The BAGC model is based on two assumptions: (1) there exists a true but unknown clustering of the vertices underlying the data; (2) vertices from the same cluster are similar to each other, while vertices from different clusters are different. In this model, the cluster label of each vertex is explicitly represented as a hidden variable. Moreover, the model enforces the intra-cluster similarity by asserting that the attribute values and edge connections of a vertex should depend on its cluster label. In particular, for vertices from the same cluster, their attribute values and edge connections should follow the common distributions that are specific to that cluster. Via the hidden clustering variable, BAGC seamlessly leverages the attribute and connection information of the vertices.

The probabilistic model essentially defines a joint probability distribution over the space of all possible clusterings and all possible attributed graphs. For a given attributed graph to be clustered, the model assigns a probability for each possible clustering of the vertices. Therefore, the clustering problem can be transformed into a standard probabilistic inference problem, i.e., to find the clustering that gives the highest probability [8]. This clustering best explains the observed attribute values and edge connections of the graph.

4.1 A Generative Process

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

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

By enumerating the values of $\mathbf{X}$ and $\mathbf{Y}$, all possible attributed graphs over V can be generated. Every instantiation of $\mathbf{X}$ and $\mathbf{Y}$ leads to a unique graph. Furthermore, if the value of $\mathbf{Z}$ is known, the tuple $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ is referred to as a clustered attributed graph. A generative process takes as input a set of vertices V, a set of attributes Λ, and the number of clusters K, and outputs a sample from all possible clustered attributed graphs $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$.

In order to generate a sample graph, BAGC needs to determine (1) an adjacency matrix $\mathbf{X} = [X_{ij}]$, (2) an attribute matrix $\mathbf{Y} = [Y_i^t]$, and (3) a clustering of vertices $\mathbf{Z} = [Z_i]$, in the following steps:

(1) It first samples the cluster label Z_i of each vertex v_i from a multinomial distribution independently. The multinomial distribution is defined as

$$p(Z_i = k|\alpha) = \alpha_k, \quad k = 1, 2, \ldots, K. \quad (14)$$

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, assume the parameter α (and also θ, ϕ below) is given. We will explain how to sample α from a Bayesian prior distribution later.

(2) Given the cluster label Z_i for vertex v_i, it then samples the at-

tribute values of this vertex. Specifically, sample the value Y_i^t of each attribute a^t from a multinomial distribution defined as

$$p(Y_i^t = m|\theta^t_{Z_i}) = \theta^t_{Z_i m}, \quad m = 1, 2, \ldots, M^t. \tag{15}$$

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

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

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

$$p(X_{ij}|\phi_{Z_i Z_j}) = \left(1 - \phi_{Z_i Z_j}\right)^{1-X_{ij}} \left(\phi_{Z_i Z_j}\right)^{X_{ij}}. \tag{16}$$

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

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

There are three parameters α, θ, and ϕ in the above generative process. BAGC takes a Bayesian approach to specify the parameter values. Instead of presuming a fixed value for each parameter, it treats α, θ, and ϕ themselves as random variables and places a *prior* distribution over them. By doing so, BAGC explicitly models the intrinsic uncertainty in the values of α, θ, and ϕ.

4.2 Model Definition

The Bayesian probabilistic model for clustered attributed graphs is defined as:

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

where

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

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

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

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

$$p(\mathbf{Y}|\mathbf{Z}, \theta) = \prod_{i=1}^{N} \prod_{t=1}^{T} p(Y_i^t|\theta^t_{Z_i}),$$

and ξ, γ, μ, ν are the hyper-parameters for sampling α, θ and ϕ.

This model makes a number of conditional independence assumptions among $\xi, \gamma, \mu, \nu, \alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$. For example, Z_i is independent of $\xi, \gamma, \mu, \nu, \theta, \phi$ given α. This is because the generation of Z_i depends only on α. Similarly, Z_i and Z_j are conditionally independent given α. This is because the cluster labels for different vertices v_i and v_j are sampled independently.

4.3 Model-based Clustering

The Bayesian model shown above 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 [8] of the clustering $\mathbf{Z}$ conditioning on $\mathbf{X}, \mathbf{Y}$. That is to find

$$\mathbf{Z}^\star = \arg\max_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \mathbf{Y}), \tag{17}$$

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

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

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

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

$$p(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) = \iiint p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y}) \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi, \tag{18}$$

where

$$p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y}) = \frac{p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z})}{\sum_{\mathbf{Z}} \iiint p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z}) \mathrm{d}\alpha \mathrm{d}\theta \mathrm{d}\phi}. \tag{19}$$

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

4.4 A Variational Algorithm

An efficient variational algorithm is developed to solve the probabilistic inference problem. The basic idea is to approximate the distribution $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$ defined in Eq. (19) using a

variational distribution $q(\alpha, \theta, \phi, \mathbf{Z})$ that is tractable for the maximization over $\mathbf{Z}$ and integration over α, θ, ϕ in Eqs. (17) and (18).

Specifically, BAGC restricts the variational distribution to a family of distributions that factorize as follows:

$$q(\alpha, \theta, \phi, \mathbf{Z}) = q(\alpha)q(\theta)q(\phi)\prod_i q(Z_i). \tag{20}$$

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

$$\begin{aligned}
\mathbf{Z}^\star &= \arg\max_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) \\
&= \arg\max_{\mathbf{Z}} \iiint p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})\mathrm{d}\alpha\mathrm{d}\theta\mathrm{d}\phi \\
&\approx \arg\max_{\mathbf{Z}} \iiint q(\alpha, \theta, \phi, \mathbf{Z})\mathrm{d}\alpha\mathrm{d}\theta\mathrm{d}\phi \\
&= \arg\max_{\mathbf{Z}} \iiint q(\alpha)q(\theta)q(\phi)\prod_i q(Z_i)\mathrm{d}\alpha\mathrm{d}\theta\mathrm{d}\phi \\
&= \arg\max_{\mathbf{Z}} \prod_i q(Z_i) \\
&= \left[\arg\max_{Z_1} q(Z_1), \ldots, \arg\max_{Z_N} q(Z_N)\right].
\end{aligned} \tag{21}$$

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

To measure the distance between a variational distribution $q(\alpha, \theta, \phi, \mathbf{Z})$ and the true posterior $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$, BAGC adopts the Kullback-Leibler (KL) divergence [1] that is commonly used in information theory and machine learning. It is defined as

$$\mathrm{KL}(q||p) = \sum_{\mathbf{Z}} \iiint q(\alpha, \theta, \phi, \mathbf{Z}) \log \frac{q(\alpha, \theta, \phi, \mathbf{Z})}{p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})}\mathrm{d}\alpha\mathrm{d}\theta\mathrm{d}\phi. \tag{22}$$

The problem is thus to find the optimal variational parameters that minimize the KL divergence. However, this optimization problem is infeasible because the KL divergence involves the term $p(\alpha, \theta, \phi, \mathbf{Z}|\mathbf{X}, \mathbf{Y})$, which is exactly what needs to be approximated in the first place.

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

$$\widetilde{L}(q) = \sum_{\mathbf{Z}} \iiint q(\alpha, \theta, \phi, \mathbf{Z}) \log \frac{p(\alpha, \theta, \phi, \mathbf{X}, \mathbf{Y}, \mathbf{Z})}{q(\alpha, \theta, \phi, \mathbf{Z})}\mathrm{d}\alpha\mathrm{d}\theta\mathrm{d}\phi. \tag{23}$$

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

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

To maximize the objective function $\widetilde{L}(q)$, we first characterize its stationary points. Based on the stationary point equations, an iterative procedure is designed for maximizing $\widetilde{L}(q)$. Reference [15] shows that the iterative maximization procedure is guaranteed to converge with a finite number of iterations. In particular, it will converge to a local maximum of $\widetilde{L}(q)$.

5. Other Related Works

There are some other related works on clustering attributed graphs. We briefly introduce them here.

Neville et al. [6] studied attributed graph clustering and proposed a weighted adjacency matrix as the similarity measure. The weight of each edge is defined as the number of attribute values shared by the two end vertices. They then applied three existing graph clustering algorithms, i.e., Min-Cut algorithm [4], MajorClust algorithm [11], and Spectral algorithm [10], on the weighted adjacency matrix to perform clustering. Steinhaeuser and Chawla [12] proposed to use attribute similarity as edge weights, similar as Ref. [6]. For each nominal attribute, if two connected nodes have the same value then increment the edge weight by one. For continuous attributes, to find the weight of edge $e(i, j)$, the method first normalizes each attribute to $(0, 1)$ and then takes the arithmetic difference between the pairs of attribute values to obtain a similarity score. Then communities are detected using a simple thresholding method. Given a threshold t, two nodes whose edge weight exceeds the threshold are put in the same community.

Another two works [2], [16] take a model-based approach. Reference [16] adopts a similar generative process as Ref. [15], and also proposes a probabilistic model to cluster attributed graphs. There are several major differences between Refs. [16] and [15]. First, Ref. [16] targets on continuous attributes, and cannot deal with categorical attributes. Second, that work treats model parameters as fixed values, while Ref. [15] takes a Bayesian treatment on the model parameters. Last, Ref. [15] assumes the number of clusters K is given, but Ref. [16] treats the number of clusters as an unknown parameter of the statistical model, and uses the Integrated Classification Likelihood (ICL) criterion to choose the optimal number of classes. Reference [2] introduces a novel Bayesian framework for hybrid community discovery in graphs. The proposed framework, HCDF (short for *Hybrid Community Discovery Framework*), can effectively incorporate hints from a number of other community detection algorithms and produce results that outperform the constituent parts. Reference [2] applies the Latent Dirichlet Allocation (LDA) as the core Bayesian method for community detection. But it deals with edge attributes rather than vertex attributes.

Tian et al. [13] also study attributed graphs. But their goal is to summarize large graphs by grouping nodes, so that vertices in one group share the same attribute values and relate to vertices in another group through the same type of relationship. Reference [13] introduces two database-style operations to summarize graphs. The first operation, called *SNAP*, produces a summary graph by grouping nodes based on user-selected attributes and relationships. The summary graph contains a number of disjoint groups, each containing a set of nodes. For any two nodes in the same group, they have the same attribute values on the user-selected attributes. In addition, for any two nodes in the same group, they relate to the same set of other groups through user-specified relationship types.

The SNAP operation produces a grouping in which nodes of each group are homogeneous with respect to user-selected at-

tributes and relationships. But homogeneity is often too restrictive in practice, as most real life graph data is subject to noise and uncertainty. Applying SNAP on noisy data can result in a large number of small groups, which is not very useful in practice. Therefore, the second operation, called *k-SNAP*, relaxes the homogeneity requirement for the relationships, based on the following observation. For each pair of groups in the result of the SNAP operation, if there is a group relationship between the two, then every node in both groups participates in this group relationship. On the other hand, if there is no group relationship between two groups, then absolutely no relationship connects any nodes across the two groups. However, in reality, if most (not all) nodes in the two groups participate in the group relationship, it is often a good indication of a strong relationship between the two groups. Likewise, it is intuitive to mark two groups as being weakly related if only a small fraction of nodes are connected between these groups. Based on these observations, the homogeneity requirement for the relationships can be relaxed, by not requiring that every node participates in a group relationship. k-SNAP allows users to control the resolutions of summaries by specifying the required number of groups k, and provides the drill-down and roll-up abilities to navigate through summaries with different resolutions.

The SNAP and k-SNAP operations achieve homogeneous attribute values within clusters, but do not enforce dense topological connection within clusters. Therefore, the generated groups will have very low connectivity within themselves.

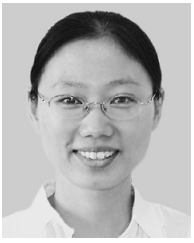
6. Conclusions

With the proliferation of rich information available for real-world objects, attributed graphs are becoming increasingly popular and important. In this paper, we present a brief overview of the state-of-the-art methods on clustering attributed graphs. These methods solve the clustering problem using different approaches, including the distance-based approach and the model-based approach. The attributed graph clustering methods can provide us a better understanding and management of the large attributed graphs arising in different applications domains, by considering both the node attribute similarity and the topological connectivity. The current research outcomes are quite encouraging, and we believe there are still many research problems that can be explored along this direction in the future.

Acknowledgments This work was supported by the Hong Kong Research Grants Council (RGC) General Research Fund (GRF) Project No.CUHK 419109 and 411211.

References

[1] Cover, T.M. and Thomas, J.A.: *Elements of information theory*, Wiley-Interscience (1991).
[2] Henderson, K., Eliassi-Rad, T., Papadimitriou, S. and Faloutsos, C.: HCDF: A Hybrid Community Discovery Framework, *SDM*, pp.754–765 (2010).
[3] Hinneburg, A. and Keim, D.A.: An Efficient Approach to Clustering in Large Multimedia Databases with Noise, *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining* (*KDD'98*), New York, NY, pp.58–65 (1998).
[4] Karger, D.R.: Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm, *SODA*, pp.21–30 (1993).
[5] Kaufman, L. and Rousseeuw, P.J.: Clustering by means of medoids, *Statistical Data Analysis based on the L1 Norm*, pp.405–416 (1987).
[6] Neville, J., Adler, M. and Jensen, D.: Clustering Relational Data Using Attribute and Link Information, *Text Mining and Link Analysis Workshop, IJCAI*, pp.689–698 (2003).
[7] Newman, M.E.J. and Girvan, M.: Finding and evaluating community structure in networks, *Phys. Rev. E*, Vol.69, 026113 (2004).
[8] Pearl, J.: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers (1988).
[9] Satuluri, V. and Parthasarathy, S.: Scalable graph clustering using stochastic flows: Applications to community discovery, *Proc. 2009 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining* (*KDD'09*), Paris, France, pp.737–745 (2009).
[10] Shi, J. and Malik, J.: Normalized cuts and image segmentation, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.22, No.8, pp.888–905 (2000).
[11] Stein, B. and Niggemann, O.: On the nature of structure and its identification, *25th Workshop on Graph Theory, Lecture Notes in Computer Science*, Springer-Verlag (1999).
[12] Steinhaeuser, K. and Chawla, N.V.: Community Detection in a Large Real-World Social Network, *Social Computing, Behavioral Modeling, and Prediction*, pp.168–175 (2008).
[13] Tian, Y., Hankins, R.A. and Patel, J.M.: Efficient aggregation for graph summarization, *Proc. 2008 ACM-SIGMOD Int. Conf. Management of Data* (*SIGMOD'08*), Vancouver, Canada, pp.567–580 (2008).
[14] Xu, X., Yuruk, N., Feng, Z. and Schweiger, T.A.J.: SCAN: A structural clustering algorithm for networks, *Proc. 2007 Int. Conf. Knowledge Discovery and Data Mining* (*KDD'07*), San Jose, CA, pp.824–833 (2007).
[15] Xu, Z., Ke, Y., Wang, Y., Cheng, H. and Cheng, J.: A model-based approach to attributed graph clustering, *SIGMOD*, pp.505–516 (2012).
[16] Zanghi, H., Volant, S. and Ambroise, C.: Clustering based on random graph model embedding vertex features, *Pattern Recognition Letters*, Vol.31, No.9, pp.830–836 (2010).
[17] Zhou, Y., Cheng, H. and Yu, J.X.: Graph clustering based on structural/attribute similarities, *VLDB*, pp.718–729 (2009).
[18] Zhou, Y., Cheng, H. and Yu, J.X.: Clustering Large Attributed Graphs: An Efficient Incremental Approach, *ICDM*, pp.689–698 (2010).



Hong Cheng is an Assistant Professor in the Department of Systems Engineering and Engineering Management at the Chinese University of Hong Kong. She received her Ph.D. degree from University of Illinois at Urbana-Champaign in 2008. Her research interests include data mining, database systems, and machine learning. She has published over 50 research papers in international conferences and journals, including SIGMOD, VLDB, SIGKDD, ICDE, IEEE Transactions on Knowledge and Data Engineering, ACM Transactions on Knowledge Discovery from Data, and Data Mining and Knowledge Discovery, and received research paper awards at ICDE'07, SIGKDD'06 and SIGKDD'05. She is a finalist for the 2009 SIGKDD Doctoral Dissertation Award. She is a recipient of the 2010 Vice-Chancellor's Exemplary Teaching Award at the Chinese University of Hong Kong.

Jeffrey Xu Yu is a Professor in the Department of Systems Engineering and Engineering Management, the Chinese University of Hong Kong. His current main research interests include graph mining, graph query processing, graph pattern matching, and keywords search in relational databases. Dr. Yu served as an Information Director and a member in ACM SIGMOD executive committee (2007–2011), and an associate editor of IEEE Transactions on Knowledge and Data Engineering (2004–2008). He serves in VLDB Journal editorial board and the International Journal of Cooperative Information Systems, and serves as an associate editor of the World Wide Web (WWW) Journal, the Journal of Information Processing (JIP), and the journal on Health Information Science and Systems (HISS). He is an ACM member and an IEEE senior member.