Graph Clustering Based on Structural/Attribute Similarities

Yang Zhou Hong Cheng Jeffrey Xu Yu
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
{zhouy, hcheng, yu}@se.cuhk.edu.hk

ABSTRACT

The goal of graph clustering is to partition vertices in a large graph into different clusters based on various criteria such as vertex connectivity or neighborhood similarity. Graph clustering techniques are very useful for detecting densely connected groups in a large graph. Many existing graph clustering methods mainly focus on the topological structure for clustering, but largely ignore the vertex properties which are often heterogenous. In this paper, we propose a novel graph clustering algorithm, *SA-Cluster*, based on both structural and attribute similarities through a unified distance measure. Our method partitions a large graph associated with attributes into k clusters so that each cluster contains a densely connected subgraph with homogeneous attribute values. An effective method is proposed to automatically learn the degree of contributions of structural similarity and attribute similarity. Theoretical analysis is provided to show that SA-Cluster is converging. Extensive experimental results demonstrate the effectiveness of SA-Cluster through comparison with the state-of-the-art graph clustering and summarization methods.

1. INTRODUCTION

Clustering is a useful and important unsupervised learning technique widely studied in literature [1, 6, 9, 17]. The general goal of clustering is to group similar objects into one cluster while partitioning dissimilar objects into different clusters. Clustering has broad applications including the analysis of business and financial data, biological data, time series data, spatial data, and so on.

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 [16, 19, 26]. Clustering on a large graph aims to partition the graph into several densely connected components. Typical applications of graph clustering include community detection in social networks, identification of functional related protein modules in large protein-protein interaction networks, *etc.*. Many existing graph clustering methods mainly focus on the topological structure of a graph so that each partition achieves a cohesive internal structure. Such methods include clustering based on normalized cut [19], modularity [16] or structural density [26]. On the other hand, one recent graph summarization method [22] aims to partition the graph according to attribute similarity, so that nodes with the same attribute values are grouped into one partition.

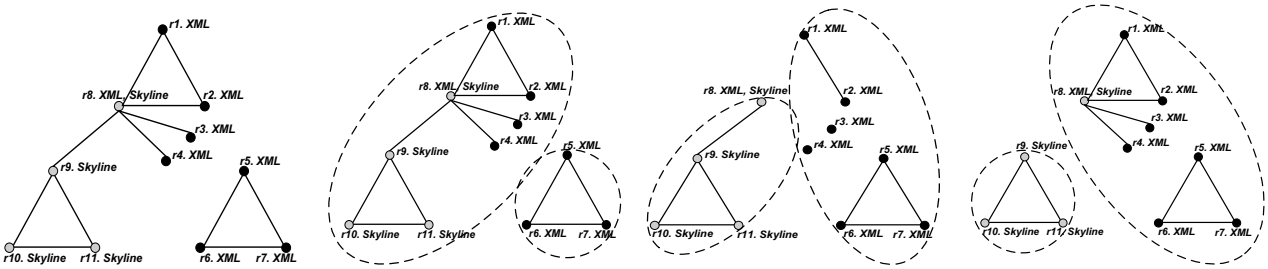
A major difference between graph clustering and traditional relational data clustering is that, graph clustering measures vertex closeness based on connectivity (*e.g.*, the number of possible paths between two vertices) and structural similarity (*e.g.*, the number of common neighbors of two vertices); while relational data clustering measures distance mainly based on attribute similarity (*e.g.*, Euclidian distance between two attribute vectors).

In many real applications, both the graph topological structure and the vertex properties are important. For example, in a social network, vertex properties describe roles of a person while the topological structure represents relationships among a group of people. The graph clustering and summarization approaches mentioned above consider only one aspect of the graph properties but ignore the other. As a result, the clusters thus generated would either have a rather random distribution of vertex properties within clusters, or have a rather loose intra-cluster structure. An ideal graph clustering should generate clusters which have a *cohesive* intra-cluster structure with *homogeneous* vertex properties, by balancing the structural and attribute similarities. Let us look at an example as follows.

Figure 1 (a) shows an illustrating example of 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 primary 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 could 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/Attribute Clustering**. Figure 1 (d) shows the

VLDB '09, August 24-28, 2009, Lyon, France



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

Figure 1: A Coauthor Network Example with an Attribute "Topic"

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. This is the result we want to achieve in this work.

The problem we study in this paper is to cluster a large-scale graph associated with attributes based on both structural and attribute similarities. The goal is to partition the graph into k clusters with cohesive intra-cluster structures and homogeneous attribute values. The problem is quite challenging because structural and attribute similarities are two seemingly independent, or even conflicting goals – in our example, authors who collaborate with each other may have different properties, such as *research topics, positions held, and prolific values*; while authors who work on the same topics may come from different groups, so they never collaborate and may even compete. It is not clear how to balance these two objectives. A possible solution is to design a distance function between two vertices v_i and v_j as

$$d(v_i, v_j) = \alpha \cdot d_S(v_i, v_j) + \beta \cdot d_A(v_i, v_j) \quad (1)$$

where $d_S(v_i, v_j)$ and $d_A(v_i, v_j)$ measure the structural distance and attribute distance respectively, while α and β are the weighting factors. Although this method is simple, it is hard to set/tune the parameters as well as interpret the weighted distance function. For our example in Figure 1 (a), it is not clear to a user whether the weight of coauthor relationship should be larger or smaller than the weight of research topic similarity. It is even harder for the user to decide the weights quantitatively.

In this paper, we seek to integrate the structural and attribute similarities into a unified framework through graph augmentation. We insert a set of *attribute vertices* to a graph G. An attribute vertex v_{jk} represents an attribute-value pair (a_j, a_{jk}). If a vertex v_i has the value a_{jk} on attribute a_j, an *attribute edge* is added between v_i and v_{jk}. To avoid confusion, original vertices are called *structure vertices* and original edges are called *structure edges*. With such graph augmentation, we express the attribute similarity as vertex vicinity in the graph: two vertices which share an attribute value are connected by a common attribute vertex. In the augmented graph, two structure vertices v_i and v_j are close either if they are connected through many other structure vertices, or if they share many common attribute vertices as neighbors, or both. Then we are able to design a distance measure which estimates the pairwise vertex closeness in the graph through both structure and attribute edges. In this paper, we propose to use the *neighborhood random walk model* to estimate the vertex closeness on the augmented graph. We could then perform graph clustering based on the random walk distance. In this problem formulation, we identify the following challenges.

1. **Adjust the degree of contributions of structural and attribute similarities.** A structure edge and an attribute edge may have different importance in random walk paths. Different attributes may also have different contributions in random walk distance due to their different clustering tendencies. For example, *research topic* could be a good attribute for grouping researchers with similar topics, while attributes like *affiliation* and *gender* may not have good clustering tendencies. To model the degree of contributions of attributes, we assign a weight to each attribute. Then we need a mechanism to differentiate the weights of different attribute edges and need a learning algorithm to automatically adjust the weights as we partition the graph.

2. **Guarantee clustering convergence.** We will design a clustering objective function and aim to improve it towards convergence. But as the attribute edge weights are adjusted, the random walk distances are affected. So if we interleave the graph clustering process and attribute edge weight adjustment for progressive refinement of the cluster quality, will the clustering process converge?

We will address the challenges listed above and propose our graph clustering algorithm based on a unified neighborhood random walk distance. The main contributions of this paper are summarized below.

1. We study the problem of clustering attributed graphs. We propose a unified distance measure to combine structural and attribute similarities. Attribute vertices and edges are added to the original graph to connect vertices which share attribute values. A neighborhood random walk model is used to measure the vertex closeness on the augmented graph through structure edges and attribute edges.

2. Theoretical analysis is provided to quantify the contribution of attribute similarity to the unified random walk distances for measuring vertex closeness.

3. We propose a weight self-adjustment method to learn the degree of contributions of different attributes in random walk distances. We also prove that the edge weights are adjusted towards the direction of clustering convergence.

4. We perform extensive evaluation of our proposed clustering approach by using real large graphs, demonstrating that our method is able to partition the graph into high-quality clusters with cohesive structures and homogeneous attribute values. In addition, we show through experiments that our clustering algorithm converges very quickly.

The rest of this paper is organized as follows. Section 2 introduces the preliminary concepts and formulates the attributed graph clustering problem. Section 3 presents a unified framework based on neighborhood random walk to integrate structural and attribute similarities. We propose an adaptive clustering algorithm for the attributed graph in Section 4. Section 5 presents extensive experimental results, followed by related work on graph clustering and graph mining in Section 6. Finally, Section 7 concludes the paper.

2. PROBLEM STATEMENT

An *attributed graph* is denoted as $G = (V, E, \Lambda)$, where V is the set of vertices, E is the set of edges, and $\Lambda = \{a_1, ..., a_m\}$ is the set of m attributes associated with vertices in V for describing vertex properties. Each vertex $v_i \in V$ is associated with an attribute vector $[a_1(v_i), ..., a_m(v_i)]$ where $a_j(v_i)$ is the attribute value of vertex v_i on attribute a_j. We denote the size of the vertex set as $|V| = N$.

Attributed graph clustering is to partition an attributed graph G into k disjoint subgraphs $G_i = (V_i, E_i, \Lambda)$, where $V = \bigcup_{i=1}^{k} V_i$ and $V_i \bigcap V_j = \phi$ for any $i \neq j$. A desired clustering of attributed graph should achieve a good balance between the following two properties: (1) vertices within one cluster are close to each other in terms of structure, while vertices between clusters are distant from each other; and (2) vertices within one cluster have similar attribute values, while vertices between clusters could have quite different attribute values.

In the attributed graph clustering problem, there are two main issues: (1) a distance measure, and (2) a clustering algorithm. We will discuss these two issues in the following sections.

3. DISTANCE IN AN ATTRIBUTED GRAPH

3.1 Structural Closeness Measure

In a large graph G, some vertices are close to each other while some other vertices are far apart based on connectivity. If there are multiple paths connecting two vertices v_i and v_j, then they are close. On the other hand, if there are very few or no paths between v_i and v_j, then they are far apart. In this paper, we use neighborhood random walk distances to measure vertex closeness.

Definition 1. [Neighborhood Random Walk Distance] Let P be the $N \times N$ transition probability matrix of a graph G. Given l as the length that a random walk can go, $c \in (0, 1)$ as the restart probability, the neighborhood random walk distance $d(v_i, v_j)$ from v_i to v_j is defined as

$$d(v_i, v_j) = \sum_{\substack{\tau: v_i \rightsquigarrow v_j \\ length(\tau) \leq l}} p(\tau) c (1-c)^{length(\tau)} \quad (2)$$

where τ is a path from v_i to v_j whose length is $length(\tau)$ with transition probability $p(\tau)$.

The matrix form of the neighborhood random walk distance is

$$R^l = \sum_{\gamma=1}^{l} c(1-c)^{\gamma} P^{\gamma} \quad (3)$$

Here, P is the transition probability matrix for graph G, and R is the neighborhood random walk distance matrix. According to Eq.(3), the recursive form of the random walk distance matrix is

$$R^l = \sum_{\gamma=1}^{l} c(1-c)^{\gamma} P^{\gamma} = c(1-c)^l P^l + R^{l-1} \quad (4)$$

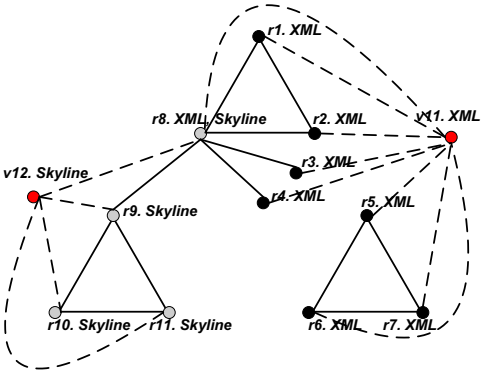


Figure 2: Attribute Augmented Graph with Topics

Then the structural closeness between two vertices v_i and v_j is

$$d_S(v_i, v_j) = R^l(i, j)$$

3.2 A Unified Distance Measure

In an attributed graph, each vertex is associated with a set of attributes $\Lambda = \{a_1, ..., a_m\}$, which describe the properties of the vertex. A straightforward way to combine structural and attribute similarities is to to use a weighted distance function as in Eq.(1) with two weighting factors α and β. Although this method is very simple, it is not easy to set the parameters and interpret the weighted distance function. Instead of modeling structural similarity and attribute similarity separately, we propose to use a unified distance measure based on the neighborhood random walk model to combine the structural closeness and attribute similarity. Before defining the distance measure, we define an *attribute augmented graph*.

Definition 2. [Attribute augmented graph] Given an attributed graph $G = (V, E, \Lambda)$ where $\Lambda = \{a_1, ..., a_m\}$. The domain (the set of possible values) of attribute a_i is $Dom(a_i) = \{a_{i1}, ..., a_{in_i}\}$ where $|Dom(a_i)| = n_i$. An attribute augmented graph is denoted as $G_a = (V \cup V_a, E \cup E_a)$ where $V_a = \{v_{ij}\}_{i=1,j=1}^{m,\ n_i}$ is the set of attribute vertices. An attribute vertex $v_{ij} \in V_a$ represents that attribute i takes the j^{th} value. An edge $(v_i, v_{jk}) \in E_a$ iff $a_j(v_i) = a_{jk}$, *i.e.*, the structure vertex v_i takes a value of a_{jk} on attribute a_j. An edge $(v_i, v_j) \in E$ is called a structure edge and an edge $(v_i, v_{jk}) \in E_a$ is called an attribute edge.

In the attribute augmented graph, we add a set of "dummy" vertices V_a where each dummy vertex represents an $< attribute, value >$ pair. An edge is added between a vertex $v_i \in V$ and a vertex $v_{jk} \in V_a$ if vertex v_i takes the k^{th} value on attribute j. Since each vertex $v_i \in V$ has m attribute values, there are totally $|V| \cdot m$ attribute edges added to the original graph G. 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 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.

We propose to use the neighborhood random walk model on the attribute augmented graph G_a to compute a unified distance between vertices in V. One important difference between the random walk on the attribute augmented graph G_a and that on the original graph G is that, if two vertices $v_i, v_j \in V$ have the same attribute value a_{kp} on attribute a_k, they will have a new common

$$P_A = \begin{array}{c} \\ r_1 \\ r_2 \\ \dots \\ \dots \\ r_{10} \\ r_{11} \\ v_{11} \\ v_{12} \end{array} \begin{array}{c} \begin{array}{ccccccc} r_1 & r_2 & \dots & r_{10} & r_{11} & v_{11} & v_{12} \end{array} \\ \begin{pmatrix} 0 & 1/3 & \dots & 0 & 0 & 1/3 & 0 \\ 1/3 & 0 & \dots & 0 & 0 & 1/3 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 1/3 & 0 & 1/3 \\ 0 & 0 & \dots & 1/3 & 0 & 0 & 1/3 \\ 1/8 & 1/8 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & \dots & 1/4 & 1/4 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 3: Transition Probability Matrix of the Attribute Augmented Graph Example

neighbor, *i.e.*, the attribute vertex $v_{kp} \in V_a$, thus there is a random walk path between v_i and v_j through v_{kp}. Obviously, the more attribute values two vertices share, the more random walk paths exist between the pair of vertices. Since $|Dom(a_i)| = n_i, \forall a_i \in \Lambda$, $|V \cup V_a| = |V| + |V_a| = N + \sum_{i=1}^{m} n_i$. The transition matrix P_A of the attribute augmented graph is a $|V \cup V_a|$ by $|V \cup V_a|$ matrix. The transition probability $P_A(v_i, v_j)$ 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 m 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 $a_1, a_2, \dots, a_m$ have an edge weight of $\omega_1, \omega_2, \dots, \omega_m$, respectively. Therefore, the transition probability from vertex v_i to vertex v_j through a structure edge is

$$p_{v_i,v_j} = \begin{cases} \dfrac{\omega_0}{|N(v_i)| * \omega_0 + \omega_1 + \omega_2 + \dots + \omega_m}, & if (v_i, v_j) \in E \\ 0, & otherwise \end{cases} \quad (5)$$

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

$$p_{v_i,v_{jk}} = \begin{cases} \dfrac{\omega_j}{|N(v_i)| * \omega_0 + \omega_1 + \omega_2 + \dots + \omega_m}, & if (v_i, v_{jk}) \in E_a \\ 0, & otherwise \end{cases} \quad (6)$$

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

$$p_{v_{ik},v_j} = \begin{cases} \dfrac{1}{|N(v_{ik})|}, & if (v_{ik}, v_j) \in E_a \\ 0, & otherwise \end{cases} \quad (7)$$

Finally, the transition probability between two attribute vertices v_{ip} and v_{jq} is 0 since there is no edge between two attribute vertices.

$$p_{v_{ip},v_{jq}} = 0, \forall v_{ip}, v_{jq} \in V_a \quad (8)$$

Since every vertex has a value on attribute a_i, the following constraint should be satisfied as well.

$$\sum_{k=1}^{n_i} |N(v_{ik})| = |V|, \ 1 \le i \le m \quad (9)$$

Combining Eqs.(5)–(8), the transition probability matrix P_A of an attribute augmented graph G_a can be computed. A transition probability matrix is shown in Figure 3 on the set of structure vertices (author) r_1–r_{11} and attribute vertices (topic) v_{11}–v_{12} for our example in Figure 2. We initialize $\omega_0 = \omega_1 = 1.0$ as the initial values. A mechanism on how to automatically adjust these edge weights will be introduced in Section 4.4.

Definition 3. [Unified Neighborhood Random Walk Distance] 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 restart probability, the unified neighborhood random walk distance $d(v_i, v_j)$ from v_i to v_j in G_a is defined as follows:

$$d(v_i, v_j) = \sum_{\substack{\tau: v_i \rightsquigarrow v_j \\ length(\tau) \le l}} p_A(\tau) c (1-c)^{length(\tau)} \quad (10)$$

where τ is a path from v_i to v_j whose length is $length(\tau)$ with transition probability $p_A(\tau)$.

The matrix form of the neighborhood random walk distance is

$$R_A^l = \sum_{\gamma=1}^{l} c(1-c)^{\gamma} P_A^{\gamma} \quad (11)$$

Here, P_A is the transition probability matrix for graph G_a, and R_A is the neighborhood random walk distance matrix. For the author-topic example, given P_A in Figure 3, $c = 0.2$ and $l = 2$, then $R_A^2(r_1, r_5) = 0.005$, which means authors r_1 and r_5 are reachable within 2 steps with 0.005 probability through the topic vertex *XML*. On the other hand, without attribute edges based on topic, the random walk distance on the original graph G is $R^l(r_1, r_5) = 0$ for an arbitrary l because r_1 and r_5 are not reachable from each other in G.

For ease of presentation, we simplify the representation of P_A as

$$P_A = \begin{bmatrix} P_V & A \\ B & O \end{bmatrix}$$

where P_V is an $N \times N$ matrix representing the transition probabilities defined by Eq.(5); $A = [A_1, A_2, ..., A_N]^T$ is a $|V| \times |V_a|$ matrix representing the transition probabilities defined by Eq.(6); $B = [B_1, B_2, ..., B_N]$ is a $|V_a| \times |V|$ matrix representing the transition probabilities defined by Eq.(7); and O is a $|V_a| \times |V_a|$ matrix with all 0s. We can get an $N \times N$ matrix C as the product of A and B where

$$C(i, j) = A_i B_j = \sum_{k=1}^{m} A(v_i, v_{kp}) \cdot B(v_{kp}, v_j) \quad (12)$$

v_{kp} is the value taken by vertex v_i on attribute a_k, $k = 1, ..., m$. If vertex v_j has a different value on a_k other than v_{kp}, then $B(v_{kp}, v_j) = 0$, thus $A(v_i, v_{kp}) \cdot B(v_{kp}, v_j) = 0$. C reflects the probability of random walk paths from one structure vertex to another in two steps by going through attribute vertices. For two arbitrary structure vertices v_i and v_j, the more attribute vertices they share as neighbors, the larger $C(i, j)$ is. When v_i and v_j have no common values on any attribute, $C(i, j) = 0$ is minimum. On the other hand, $C(i, j)$ is maximum if v_i and v_j share the same value for each attribute.

Lemma 1. *Given P_A and a positive integer l, P_A^l can be represented as*

$$P_A^l = \begin{bmatrix} T_l & T_{l-1}A \\ BT_{l-1} & BT_{l-2}A \end{bmatrix}$$

where $T_l = P_V T_{l-1} + CT_{l-2}$.

Proof. We will prove it by induction. When $l = 1$,

$$P_A^1 = \begin{bmatrix} P_V & A \\ B & O \end{bmatrix}$$

and $T_1 = P_V$. When $l = 2$,

$$P_A^2 = \begin{bmatrix} P_V^2 + C & P_V A \\ BP_V & BA \end{bmatrix}$$

and $T_2 = P_V^2 + C$. When $l = k$, assume that we have

$$T_k = P_V T_{k-1} + CT_{k-2}$$

When $l = k + 1$,

$$P_A^{k+1} = P_A \cdot P_A^k = \begin{bmatrix} P_V & A \\ B & O \end{bmatrix} \times \begin{bmatrix} T_k & T_{k-1}A \\ BT_{k-1} & BT_{k-2}A \end{bmatrix}$$

$$= \begin{bmatrix} P_V T_k + ABT_{k-1} & P_V T_{k-1}A + ABT_{k-2}A \\ BT_k & BT_{k-1}A \end{bmatrix}$$

$$= \begin{bmatrix} P_V T_k + CT_{k-1} & (P_V T_{k-1} + CT_{k-2})A \\ BT_k & BT_{k-1}A \end{bmatrix} = \begin{bmatrix} T_{k+1} & T_k A \\ BT_k & BT_{k-1}A \end{bmatrix}$$

According to the recursive definition of T_l, T_l can be rewritten in the form of $\sum \prod_{m_i \geq 0, n_i \geq 0} (P_V^{m_i} C^{n_i})$.

THEOREM 1. *Given two vertices $v_p, v_q \in V$, assume that v_p and v_q have exactly the same connectivity in G, i.e., $\forall v_j \in V$, $(v_j, v_p) \in E$, iff $(v_j, v_q) \in E$. Given another vertex $v_i \in V$, assume that v_i shares more attribute values with v_p than with v_q. We further assume that, $\forall v_j \neq v_i, v_p, v_q$, v_j shares the same number of attribute values with v_p as with v_q. Then for an arbitrary random walk length l, $P_A^l(v_i, v_p) > P_A^l(v_i, v_q)$.*

Proof. See Appendix.

THEOREM 2. *Given two vertices $v_p, v_q \in V$, assume that v_p and v_q have exactly the same connectivity in G, i.e., $\forall v_j \in V$, $(v_j, v_p) \in E$, iff $(v_j, v_q) \in E$. Given another vertex $v_i \in V$, assume that v_i shares more attribute values with v_p than with v_q. We further assume that, $\forall v_j \neq v_i, v_p, v_q$, v_j shares the same number of attribute values with v_p as with v_q. Then for an arbitrary random walk length l, $R_A^l(v_i, v_p) > R_A^l(v_i, v_q)$.*

Proof. According to Theorem 1, we have $P_A^l(v_i, v_p) > P_A^l(v_i, v_q)$. According to Eq.(11), R_A^l is a linear sum of P_A^γ with positive coefficients $c(1-c)^\gamma$, $\gamma = 1, ..., l$. Then we have $R_A^l(v_i, v_p) > R_A^l(v_i, v_q)$ for an arbitrary l.

Theorem 2 demonstrates that attribute similarity increases the closeness between two vertices in an attribute augmented graph through neighborhood random walks. Based on this neighborhood random walk model in an attribute augmented graph, we unify the structural and attribute similarities into one random walk distance measure effectively.

4. CLUSTERING ALGORITHM

Our clustering framework is to partition an attributed graph G based on both structural and attribute similarities through a unified neighborhood random walk model on the attribute-augmented graph G_a of G. In this section, we will address the challenges in the clustering process by answering the following questions. Based on our answers to the questions, we propose an adaptive clustering algorithm on an attributed graph.

1. Given the unified random walk distances to measure the similarity between vertices, what kind of clustering approach shall we use?

2. We assign a weight ω_0 to structure edges and ω_i to attribute edges on attribute a_i. Different types of edges may have different degree of contributions in calculating the unified random walk distances, thus we can assume $\omega_0 \neq \omega_1 \neq ... \neq \omega_m$. Then the question is, can we learn the weights adaptively in the clustering process and how [1]?

3. What should be the objective function for clustering? Since the random walk distances are affected as the weights are updated, will the objective function converge as we iteratively adjust the weights $\{\omega_1, ..., \omega_m\}$?

Many existing graph clustering/partitioning methods focused only on either topological structures [16,19,26] or vertex attributes [22]. In our problem of clustering an attributed graph, the unified neighborhood random walks consider all possible paths through both structure edges and attribute edges. Due to the attribute edges, the attribute similarity between two vertices will bring them closer in attribute augmented graph, causing an increased random walk distance. According to the unified random walk model, two vertices are assigned to the same cluster if the random walk distance is very large [2]; while two vertices belong to different clusters if the random walk distance is very small or 0, *i.e.*, there does not exist a neighborhood random walk between them.

With the random walk distance as a pairwise similarity measure, we could ignore the original graph topological structure in the clustering process. Our clustering framework follows the *K-Medoids* clustering method [13]: we select the most centrally located point in a cluster as a centroid, and assign the rest of points to their closest centroids. In each iteration, the edge weights $\{\omega_1, ..., \omega_m\}$ are adjusted to reflect the clustering tendencies of attributes. This process is repeated until convergence. We will discuss each step in the clustering process separately.

4.1 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, we follow the motivation of identifying good initial centroids from the density point of view [10]. 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. First, we define an *influence function* as follows.

Definition 4. [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

$$f_B^{v_j}(v_i) = 1 - e^{-\frac{d(v_i, v_j)^2}{2\sigma^2}} \tag{13}$$

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 proportional to the random walk distance from v_i to v_j. The larger the random walk distance from v_i to v_j, the more influence v_i has on v_j.

Definition 5. [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

$$f_B^D(v_i) = \sum_{v_j \in V} f_B^{v_j}(v_i) = \sum_{v_j \in V} (1 - e^{-\frac{d(v_i, v_j)^2}{2\sigma^2}}) \tag{14}$$

If one vertex v_i has a large density value, it means that, either v_i connects to many vertices through multiple random walk paths, or v_i shares attribute values with many vertices.

According to the density function, we sort all vertices in the descending order of their density values. Then select the densest k vertices from the sorted list as the initial centroids $\{c_1^0, ..., c_k^0\}$.

[1] We fix $\omega_0 = 1.0$ and adjust $\{\omega_1, ..., \omega_m\}$ relative to ω_0.

[2] Different from traditional distance measures, a random walk distance measures the closeness between two vertices.

4.2 Clustering Process

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

$$c^* = argmax_{c_j^t} d(v_i, c_j^t)$$

When all vertices are assigned to some cluster, the centroid will be updated with the most centrally located vertex in each cluster. To find such a vertex, we first compute the "average point" $\overline{v_i}$ of a cluster V_i in terms of random walk distance as

$$R_A^l(\overline{v_i}, v_j) = \frac{1}{|V_i|} \sum_{v_k \in V_i} R_A^l(v_k, v_j), \forall v_j \in V \tag{15}$$

Thus $R_A^l(\overline{v_i})$ is the average random walk distance vector for cluster V_i. Then we find the new centroid c_i^{t+1} in cluster V_i as

$$c_i^{t+1} = argmin_{v_j \in V_i} \|R_A^l(v_j) - R_A^l(\overline{v_i})\| \tag{16}$$

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.

4.3 Clustering Objective Function

The objective of clustering is to maximize intra-cluster similarity and minimize inter-cluster similarity. We design our clustering objective function according to this general goal. As our distance measure is the unified random walk distance, we will maximize the intra-cluster random walk distances.

Definition 6. [Vertex Set Distance] Let V_1 and V_2 be two sets of vertices. The vertex set distance $d(V_1, V_2)$ between V_1 and V_2 is defined as

$$d(V_1, V_2) = \sum_{v_i \in V_1, v_j \in V_2} \frac{d(v_i, v_j)}{|V_1| \times |V_2|} \tag{17}$$

where $d(v_i, v_j) = R_A^l(v_i, v_j)$ is the unified neighborhood random walk distance between v_i and v_j. This formula is designed to quantitatively measure the extent of similarity between two vertex sets in a graph. When vertices from V_1 and V_2 are distant or isolated, the vertex set distance measure would be very small.

Definition 7. [Graph Clustering Objective Function] Given an attributed graph $G = (V, E, \Lambda)$, the weights $W = \{\omega_1, ..., \omega_m\}$ on attribute edges, a random walk length limit l, and the cluster number k, the goal of a graph clustering is to find k partitions $\{V_i\}_{i=1}^k$ of the graph, where V_i corresponds to the i^{th} cluster, $V_i \bigcap V_j = \phi$ and $\bigcup_{i=1}^k V_i = V$, so that the following objective function is maximized

$$O(\{V_i\}_{i=1}^k, W) = \sum_{i=1}^{k} d(V_i, V_i) \tag{18}$$

subject to $\sum_{i=1}^m \omega_i = m$ and $\omega_i \geqslant 0$, $i = 1, ..., m$.

The clustering problem can be reduced to three subproblems of (1) cluster assignment, (2) centroid update, and (3) weight adjustment, with the goal of maximizing the objective function. The first two problems have been discussed in Section 4.2 and the third one will be investigated in Section 4.4. We will first study the relationship between the weights $W = \{\omega_1, ..., \omega_m\}$ and the objective function in Eq.(18).

Theorem 3. *Given a certain partition $\{V_i^*\}_{i=1}^k$ of graph G, there exists a unique solution $W^* = \{\omega_1^*, ..., \omega_m^*\}$ which maximizes the objective function in Eq.(18).*

Proof. According to Eq.(11), we know that, $\forall v_p, v_q \in V$, $1 \leq \gamma \leq l$, $P_A^\gamma(v_p, v_q)$ is a polynomial function of multi-variable $\omega_1, ..., \omega_m$ with non-negative real coefficients. The coefficients $c(1-c)^\gamma$ in Eq.(11) are positive. Then we can infer that $R_A^l(v_p, v_q)$ is a polynomial function of $\omega_1, ..., \omega_m$ with non-negative real coefficients. Since $O(\{V_j^*\}_{j=1}^k, W)$ is the sum of multiple random walk distances, it is also a polynomial function of $\omega_1, ..., \omega_m$ with non-negative real coefficients. Assume that the polynomial expression of weight ω_i in $O(\{V_j^*\}_{j=1}^k, W)$ is expressed as $f_i(\omega_i)$. Then we can rewrite $O(\{V_j^*\}_{j=1}^k, W) = \sum_{i=1}^m f_i(\omega_i)$. Let λ be the Lagrange multiplier and $O'(\{V_j^*\}_{j=1}^k, W, \lambda)$ be the Lagrange function

$$O'(\{V_j^*\}_{j=1}^k, W, \lambda) = \sum_{i=1}^{m} f_i(\omega_i) + \lambda(\sum_{i=1}^{m} \omega_i - m) \tag{19}$$

If (W^*, λ^*) maximizes $O'(\{V_j^*\}_{j=1}^k, W, \lambda)$, W^* and λ^* are the solutions of the following two equations.

$$\frac{\partial O'(\{V_j^*\}_{j=1}^k, W, \lambda)}{\partial \omega_i} = \frac{\partial f_i(\omega_i)}{\partial \omega_i} + \lambda = g_i(\omega_i) + \lambda + c = 0, (1 \leqslant i \leqslant m) \tag{20}$$

$$\frac{\partial O'(\{V_j^*\}_{j=1}^k, W, \lambda)}{\partial \lambda} = \sum_{i=1}^{m} \omega_i - m = 0 \tag{21}$$

As $\frac{\partial f_i(\omega_i)}{\partial \omega_i}$ may generate a constant term, to illustrate this explicitly, we use $g_i(\omega_i) + c$ to represent $\frac{\partial f_i(\omega_i)}{\partial \omega_i}$ in Eq.(20). $g_i(\omega_i)$ is a polynomial function of ω_i with non-negative real coefficients.

Gauss' Fundamental Theorem of Algebra [8] states that every non-constant single-variable polynomial with complex coefficients has at least one complex root. Thus, Eq.(20) must have at least one complex root. For $\lambda = \lambda^*$, there are two possible cases, i.e., $\lambda^* \geq -c$ and $\lambda^* < -c$, to be discussed separately as follows.

***[Case 1.** $\lambda^* \geq -c$] According to Descartes' Rule of Signs [5], the number of positive roots of a polynomial function is either equal to the number of sign differences between consecutive nonzero coefficients, or less than it by a multiple of 2. When $\lambda^* + c \geq 0$, all coefficients are non-negative real numbers. Thus the number of positive roots of Eq.(20) is 0 according to Descartes' Rule of Signs. Actually we have*

$$g_i(\omega_i) + \lambda^* + c \geq \lambda^* + c \geq 0$$

for any assignments $\omega_i \geq 0$, $i = 1, ..., m$. In addition, to satisfy the constraint $\sum_{i=1}^m \omega_i = m$, there must exist at least one $\omega_i > 0$, $1 \leq i \leq m$. Then we will have

$$g_i(\omega_i) + \lambda^* + c > \lambda^* + c \geq 0$$

Thus, we prove that there are no real number solutions to Eq.(20) if $\lambda^* \geq -c$.

***[Case 2.** $\lambda^* < -c$] When $\lambda^* < -c$, the constant term $\lambda^* + c < 0$ in Eq.(20). According to Descartes' Rule of Signs, there exists one or no positive root in Eq.(20) because there is one sign difference in the equation.*

Next, we will prove that there exists at least one solution to Eq.(20) if $\lambda^* < -c$. When $\omega_i = 0$, $i = 1, ..., m$, the polynomial function $g_i(\omega_i) + \lambda^* + c = \lambda^* + c < 0$. On the other hand, as all coefficients in $g_i(\omega_i)$ are non-negative, the function value is monotonically increasing with ω_i. Hence, there must exist at least one positive number δ where $\omega_i = \delta$ so that the polynomial function is greater than 0. The polynomial function is continuous in the closed interval $[0, \delta]$.

The function value also has opposite signs at the boundary points of 0 *and* δ*. According to the Root Location Theorem or Bolzano's Theorem [2], we can conclude that there exists at least one solution to Eq.(20) in the open interval* $(0, \delta)$*.*

To summarize, for some $\lambda^* < -c$*, there exists a unique positive root* $W^* = \{\omega_1^*, ..., \omega_m^*\}$ *for Eq.(20). In addition, the function value of this positive root must be greater than that of negative roots since this function does not have any negative coefficients. Therefore, we can conclude that there exists a unique positive root which maximizes the objective function.*

4.4 Weight Self-Adjustment

The graph clustering problem through maximizing objective function with constraints in Definition 7 is a linear programming problem. An adaptive weight adjustment method is proposed to iteratively improve the objective function.

We fix $\omega_0 = 1.0$ which is the structure edge weight and iteratively adjust the attribute weights $\{\omega_1, ..., \omega_m\}$ relative to ω_0. Let $W^t = \{\omega_1^t, \ldots, \omega_m^t\}$ be the attribute weights in the t^{th} iteration. We initialize $\omega_1^0 = \omega_2^0 = \ldots = \omega_m^0 = 1.0$. We iteratively adjust 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

$$\omega_i^{t+1} = \frac{1}{2}(\omega_i^t + \triangle\omega_i^t) \tag{22}$$

To accurately determine the extent of weight increment $\triangle\omega_i$, we design 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.

$$vote_i(v_p, v_q) = \begin{cases} 1, & if\ v_p,\ v_q\ share\ the\ same\ value\ on\ a_i \\ 0, & otherwise \end{cases} \tag{23}$$

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.

$$\triangle\omega_i^t = \frac{\sum_{j=1}^{k}\sum_{v\in V_j} vote_i(c_j, v)}{\frac{1}{m}\sum_{p=1}^{m}\sum_{j=1}^{k}\sum_{v\in V_j} vote_p(c_j, v)} \tag{24}$$

The denominator in Eq.(24) ensures that the constraint $\sum_{i=1}^{m}\omega_i^{t+1} = m$ is still satisfied after weight adjustment. Then the adjusted weight is calculated as

$$\omega_i^{t+1} = \frac{1}{2}(\omega_i^t + \triangle\omega_i^t) = \frac{1}{2}(\omega_i^t + \frac{m\sum_{j=1}^{k}\sum_{v\in V_j} vote_i(c_j, v)}{\sum_{p=1}^{m}\sum_{j=1}^{k}\sum_{v\in V_j} vote_p(c_j, v)}) \tag{25}$$

The adjusted weights may be increasing, decreasing, or unchanged depending on the value of $\triangle\omega_i^t$. If $\triangle\omega_i^t > \omega_i^t$, then $\omega_i^{t+1} > \omega_i^t$, *i.e.*, attribute *i* makes an increasing contribution to the random walk distance. Similarly, if $\triangle\omega_i^t < \omega_i^t$, $\omega_i^{t+1} < \omega_i^t$. If $\triangle\omega_i^t = \omega_i^t$, $\omega_i^{t+1} = \omega_i^t$.

An important property of the weight self-adjustment mechanism is that *the weights are adjusted towards the direction of increasing the clustering objective function value*. The detailed proof is omitted due to space limit. We will briefly illustrate this property qualitatively: if a large number of vertices within clusters share an attribute value on a_i, then the weight is increased, *i.e.*, $\omega_i^{t+1} > \omega_i^t$; on the other hand, if vertices within clusters have quite different attribute values on a_i, the weight is then decreased, *i.e.*, $\omega_i^{t+1} < \omega_i^t$. There must be some weights with decreasing updates and some other weights with increasing updates, since $\sum_{i=1}^{m}\omega_i^t = m$ is a constant. Due to some increased ω_i^{t+1}, the random walk distances between vertices which share the same attribute value on a_i will be further increased. As a result, these vertices tend to be clustered into the same cluster, thus increasing the objective function which is the sum of intra-cluster random walk distances. We have proven in Theorem 3 that there exists a unique solution $W^* = \{\omega_1^*, ..., \omega_m^*\}$ which maximizes the objective function, thus we can conclude that the weight self-adjustment strategy increases the objective function towards convergence.

Algorithm 1 Attributed Graph Clustering SA-Cluster

Input: an attributed graph G, a length limit l of random walk paths, a restart probability c, a parameter σ of influence function, cluster number k.
Output: k clusters $V_1, ..., V_k$.

1: Initialize $\omega_1 = ... = \omega_m = 1.0$, fix $\omega_0 = 1.0$;
2: Calculate the unified random walk distance matrix R_A^l;
3: Select k initial centroids with highest density values;
4: Repeat until the objective function converges:
5: Assign each vertex v_i to a cluster C^* with a centroid c^* where $c^* = argmax_{c_j} d(v_i, c_j)$;
6: Update the cluster centroids with the most centrally located point in each cluster according to Eqs.(15)–(16);
7: Update weights $\omega_1, ..., \omega_m$ according to Eq.(25);
8: Re-calculate R_A^l;
9: Return k clusters $V_1, ..., V_k$.

4.5 Clustering Algorithm

By assembling different parts, our clustering algorithm SA-Cluster is presented in Algorithm 1.

THEOREM 4. *The objective function in Algorithm 1 converges to a local maximum in a finite number of iterations.*

Proof. Existing work has studied the convergence properties of the partitioning approach to clustering, such as K-Means [3]. Our clustering follows a similar clustering approach. So the cluster assignment and centroid update steps improve the objective function value. In addition, we have explained that weight adjustment also improves the objective function value. Therefore, the objective function keeps increasing and converges to a local maximum in a finite number of iterations.

In Section 5, we will demonstrate through experiments that our clustering algorithm converges very quickly on real datasets.

Another issue on the SA-Cluster algorithm is the computational complexity. We need to compute N^2 pairs of random walk distances between vertices in V through matrix multiplication. As $W = \{\omega_1, ..., \omega_n\}$ is updated, the random walk distances need to be recalculated. This computational complexity can be reduced with the recent research result on *fast random walk computation* [24].

5. EXPERIMENTAL STUDY

In this section, we performed extensive experiments to evaluate the performance of our algorithm SA-Cluster on real graph datasets. All experiments were done on a 2.8GHz Intel Pentium IV PC with 1GB main memory, running Windows XP. All algo-

rithms were implemented in C++ and compiled using Visual C++ 6.0 compiler, except that random walk was computed by Matlab.

5.1 Experimental Datasets

We use two real graph datasets for evaluation in our experiments.

Political Blogs Dataset The political blogs network dataset can be downloaded from http://www-personal.umich.edu/ mejn/netdata/. This dataset is a network of $1,490$ webblogs on US politics with $19,090$ hyperlinks between these webblogs. Each blog in the dataset has an attribute describing its political leaning as either *liberal* or *conservative*.

DBLP Dataset We use the DBLP Bibliography data from four research areas of database (DB), data mining (DM), information retrieval (IR) and artificial intelligence (AI) [3]. We build a coauthor graph with top $5,000$ authors and their coauthor relationships. In addition, we use two relevant attributes: *prolific* and *primary topic*. For attribute "prolific", authors with ≥ 20 papers are labeled as highly prolific; authors with ≥ 10 and < 20 papers are labeled as prolific and authors with < 10 papers are labeled as low prolific. For attribute "primary topic", we use a topic modeling approach [11, 27] to extract 100 research topics from a document collection composed of paper titles from the selected authors. Each extracted topic consists of a probability distribution of keywords which are most representative of the topic. Then each author will have one out of 100 topics as his/her primary topic.

5.2 Comparison Methods and Evaluation

- **k-SNAP** We implemented k-SNAP Top-Down [22] that groups vertices with the same attribute values into one cluster.
- **S-Cluster** This baseline clustering algorithm only considers topological structure. Random walk distance is used to measure vertex closeness according to Definition 1. Attribute similarity is ignored by setting $\omega_1 = ... = \omega_m = 0$.
- **W-Cluster** This is a fictitious clustering algorithm which combines structural and attribute similarities through a weighted function in Eq.(1). The weighting factors $\alpha = \beta = 0.5$. For a pair of vertices $v_i, v_j \in V$, $d_S(v_i, v_j)$ is the random walk distance, and $d_A(v_i, v_j)$ is their attribute similarity.
- **SA-Cluster** This is our proposed algorithm which considers both structural and attribute similarities.

Evaluation Measures We use two measures of *density* and *entropy* to evaluate the quality of clusters $\{V_i\}_{i=1}^k$ generated by different methods. The definitions are as follows.

$$density(\{V_i\}_{i=1}^k) = \sum_{i=1}^{k} \frac{|\{(v_p, v_q)|v_p, v_q \in V_i, (v_p, v_q) \in E\}|}{|E|} \quad (26)$$

$$entropy(\{V_i\}_{i=1}^k) = \sum_{i=1}^{m} \frac{\omega_i}{\sum_{p=1}^{m} \omega_p} \sum_{j=1}^{k} \frac{|V_j|}{|V|} entropy(a_i, V_j) \quad (27)$$

where

$$entropy(a_i, V_j) = -\sum_{n=1}^{n_i} p_{ijn} log_2 p_{ijn}$$

and p_{ijn} is the percentage of vertices in cluster j which have value a_{in} on attribute a_i. $entropy(\{V_i\}_{i=1}^k)$ measures the weighted entropy from all attributes over k clusters.

[3]The detailed conference list is, DB: SIGMOD, VLDB, PODS, ICDE, EDBT; DM: KDD, ICDM, SDM, PAKDD, PKDD; IR: SIGIR, CIKM, ECIR, WWW; AI: IJCAI, AAAI, UAI, NIPS.

In the experiments, we set the random walk length $l = 20$ and the restart probability $c = 0.15$.

5.3 Cluster Quality Evaluation

Figure 4 (a) shows the density comparison between the four methods on Political Blogs when we set the cluster number $k = 3, 5, 7, 9$. The density values by SA-Cluster and S-Cluster are close. They remain around 0.6 or above even when k is increasing. This demonstrates that both methods can find densely connected components. On the other hand, k-SNAP has a low density, and the density value decreases quickly when k increases. This is because k-SNAP partitions a graph without considering connectivity. The density by W-Cluster stands in between. It also gradually decreases as k increases.

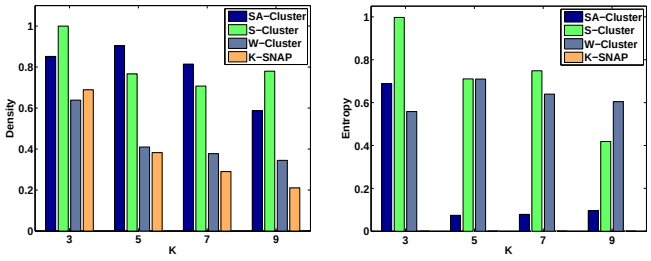
Figure 4 (b) shows the entropy comparison between the four methods on Political Blogs when we set $k = 3, 5, 7, 9$. The entropy measure is always 0 for k-SNAP, since it partitions a graph where each partition contains nodes with the same attribute value. Besides, SA-Cluster achieves a much lower entropy than S-Cluster. When $k = 5, 7$ or 9, entropy by SA-Cluster is as low as less than 0.1 while entropy by S-Cluster is still around $0.4 - 0.8$. The entropy by W-Cluster is similar to that of S-Cluster, but not as good as that of SA-Cluster or k-SNAP. As shown in Figures 4 (a) and (b), the performance of W-Cluster stands in between in terms of both density and entropy. This is because its distance function combines (or compromises) both structural and attribute similarities through a weighted function, thus it achieves a reasonable result on both criteria. On the other hand, as it is hard to set or tune the weighting factors α and β to achieve an optimal result, the performance is inferior to that of SA-Cluster.

Figures 5 (a) and (b) show density and entropy on DBLP with different k values by SA-Cluster, S-Cluster and W-Cluster. As shown in the figures, SA-Cluster achieves a much lower entropy than S-Cluster, with slightly lower density. W-Cluster achieves an even lower entropy around $0.27 - 0.40$, which is better than SA-Cluster and S-Cluster. But the density value of W-Cluster is also much lower than that of SA-Cluster and S-Cluster. We can infer that the weighted distance function in W-Cluster compromises attribute similarity and structural similarity. Since k-SNAP strictly enforces the attribute homogeneity in each cluster, with 3 values on attribute *prolific* and 100 different topics, the minimum possible number of clusters for k-SNAP is 300 in this experiment. So we ran k-SNAP with $k = 300$ and achieved a density value of 0.1244 (much lower than that of the other three methods) and an entropy of 0.

To further test the effect of attribute weights on random walk distance and on clustering quality in SA-Cluster, we manually set different weights of the attribute *political leaning* with $\omega = 0, 1, 2, 3$ on Political Blogs. Figures 6 (a) and (b) show the density and entropy respectively. $\omega = 0$ corresponds to S-Cluster. When the weight increases, the contribution of attribute similarity in random walk distance is also increasing. The clusters thus generated achieve a much lower entropy on attribute value, while the density is not sacrificed. This experiment demonstrates that introducing the attribute similarity into the clustering objective will not downgrade the intra-cluster cohesiveness.

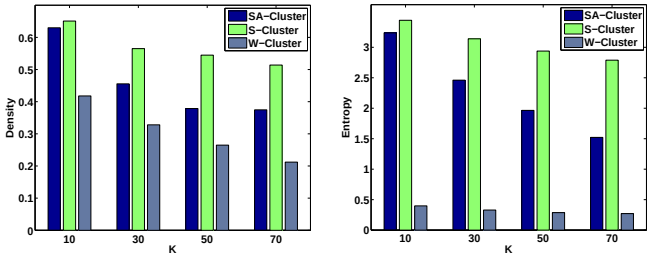
5.4 Clustering Convergence

Figures 7 (a) and (b) show the trend of clustering convergence in terms of objective function value on Political Blogs and DBLP respectively. Both figures show that the objective function keeps increasing when we iteratively perform the three tasks of cluster assignment, centroid update and attribute weight adjustment. The



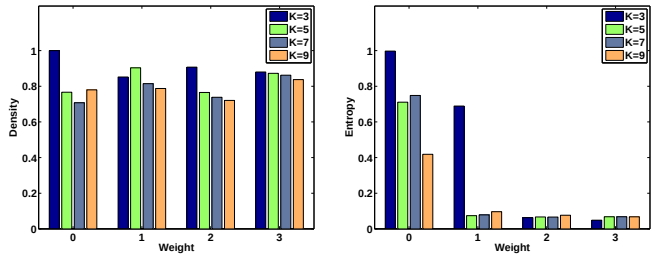
(a) density (b) entropy

Figure 4: Cluster Quality Comparison on Political Blogs



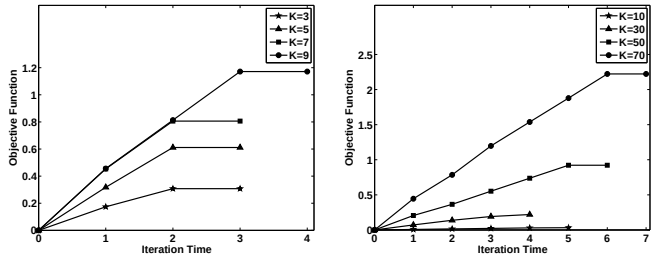
(a) density (b) entropy

Figure 5: Cluster Quality Comparison on DBLP



(a) density (b) entropy

Figure 6: Quality vs. Attribute Weights on Political Blogs



(a) Political Blogs (b) DBLP

Figure 7: Cluster Convergence

objective function converges very quickly, usually in three to seven iterations. These results are consistent with Theorem 4.

Figure 8 shows the trend of weight updates on DBLP with different k values. Here we only show the weights of attribute *prolific* since the sum of the two attribute weights is a constant. As shown in the figure, the weights are converging as the clustering process converges. Another interesting finding is that the prolific weight is decreasing as k increases. A reasonable explanation is that, when k is small, *e.g.*, $k = 10$, a cluster with many authors mixed up has a very diverse topic distribution. Thus, *topic* is not a good clustering attribute when k is small. Then the prolific weight increases for higher contribution. On the other hand, when k is large, authors with different topics can be separated more clearly. Therefore, the dependence on the prolific weight is reduced significantly.

Tables 1 and 2 show the cluster quality of SA-Cluster in terms of density (D) and entropy (E) iteration by iteration on Political Blogs and DBLP respectively, to show the effectiveness of attribute weight self-adjustment. As we can see on Political Blogs, in most cases, both density and entropy improve as the weights and clusters are computed iteratively. The biggest improvement is usually achieved at iteration 2. Similar effects can be observed in Table 2, although entropy gains more improvements while density sacrifices a little bit iteratively.

5.5 Clustering Efficiency Evaluation

In this experiment, we compare the efficiency of different clustering algorithms. Figures 9 (a) and (b) show the runtime by different methods on Political Blogs and DBLP, respectively. As we can observe, k-SNAP is the most efficient. SA-Cluster is usually several times (≤ 4) slower than S-Cluster and W-Cluster, as it iteratively computes the random walk distance and performs clustering, while S-Cluster and W-Cluster compute random walk only once. Although SA-Cluster is more expensive, the iterative refinement improves the clustering quality a lot, as demonstrated in Tables 1 and 2.

5.6 DBLP Case Study

In this experiment, we examine some details of our clustering results on DBLP data when we set $k = 70$. Table 3 shows 5 clusters of authors from database, data mining, machine learning and information retrieval areas. In each cluster, we only select the most prolific authors for ease of presentation. Actually we omit many other authors who are closely connected with the selected authors in each cluster due to their close collaborations. Table 4 shows the keywords and their probabilities of the primary topic for each cluster.

Clusters 0−4 contain five groups of authors who work on "stream data management", "database systems", "frequent pattern mining", "graphical model learning", and "information retrieval and relevance feedback", respectively. Interestingly, each cluster not only

iter#	k=3		k=5		k=7		k=9	
	D	E	D	E	D	E	D	E
1	0.78	0.84	0.81	0.25	0.77	0.08	0.69	0.09
2	0.85	0.69	0.90	0.07	0.81	0.07	0.71	0.10
3	0.85	0.69	0.90	0.07	0.81	0.07	0.59	0.09
4							0.59	0.09

Table 1: Cluster Quality In Iteration on Political Blogs

iter#	k=10		k=30		k=50		k=70	
	D	E	D	E	D	E	D	E
1	0.55	3.12	0.48	2.79	0.47	2.55	0.46	2.43
2	0.55	3.20	0.43	2.49	0.41	2.00	0.38	1.63
3	0.60	3.18	0.46	2.46	0.37	1.99	0.37	1.54
4	0.63	3.23	0.46	2.44	0.38	1.96	0.37	1.53
5	0.63	3.22			0.38	1.96	0.37	1.54
6					0.38	1.96	0.38	1.52
7							0.38	1.52

Table 2: Cluster Quality In Iteration on DBLP

Cluster 0 (DB)	Cluster 1 (DB)	Cluster 2 (DM)	Cluster 3 (ML)	Cluster 4 (IR)
Graham Cormode	François Bancilhon	Gagan Agrawal	Lawrence Carin	James Allan
Gautam Das	Alexandros Biliris	Francesco Bonchi	Honghua Dai	Javed A. Aslam
Minos N. Garofalakis	José A. Blakeley	Guozhu Dong	Pedro Domingos	Chris Buckley
Sudipto Guha	Klemens Böhm	Fosca Giannotti	Lise Getoor	Charles L. A. Clarke
Dimitrios Gunopulos	Nicolas Bruno	Jiawei Han	Zoubin Ghahramani	W. Bruce Croft
Eamonn J. Keogh	Jan Van den Bussche	Ruoming Jin	Michael I. Jordan	Norbert Fuhr
Nick Koudas	Michael J. Carey	Jinyan Li	Daphne Koller	Alexander G. Hauptmann
Flip Korn	Sharma Chakravarthy	Wagner Meira Jr.	Pat Langley	Rong Jin
Yannis Kotidis	Surajit Chaudhuri	Giuseppe Manco	Yuchang Lu	Joemon M. Jose
Xuemin Lin	Peter Dadam	Richard R. Muntz	Avi Pfeffer	David R. Karger
Hongjun Lu	David J. DeWitt	Siegfried Nijssen	Dan Roth	Mounia Lalmas
Nikos Mamoulis	César A. Galindo-Legaria	Srinivasan Parthasarathy	Padhraic Smyth	Victor Lavrenko
Rajeev Motwani	Georges Gardarin	Dino Pedreschi	Henry Tirri	Donald Metzler
S. Muthukrishnan	Goetz Graefe	Jian Pei	Volker Tresp	Alistair Moffat
M. Tamer Özsu	Theo Härder	Anthony K. H. Tung	Eric P. Xing	Jian-Yun Nie
Dimitris Papadias	Won Kim	Adriano Veloso	Kai Yu	Douglas W. Oard
Rajeev Rastogi	David Maier	Jason Tsong-Li Wang	Shipeng Yu	Jan O. Pedersen
Nick Roussopoulos	Bernhard Mitschang	Limsoon Wong		Tetsuya Sakai
Elke A. Rundensteiner	Vivek R. Narasayya	Stefan Wrobel		Mark Sanderson
Bernhard Seeger	Erich J. Neuhold	Ke Wang		Rohini K. Srihari
Kyuseok Shim	Z. Meral Özsoyoglu	Xifeng Yan		John Tait
Yufei Tao	Arnon Rosenthal	Jiong Yang		Xiaojun Wan
Michail Vlachos	Kyu-Young Whang	Philip S. Yu		S. K. Michael Wong
Jeffrey Xu Yu		Osmar R. Zaïane		Chengxiang Zhai
Carlo Zaniolo		Mohammed Javeed Zaki		Justin Zobel

Table 3: Clusters of Authors from DBLP

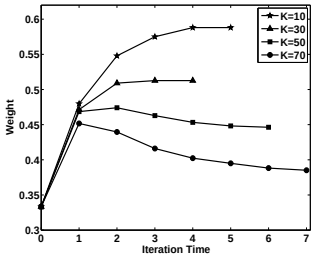
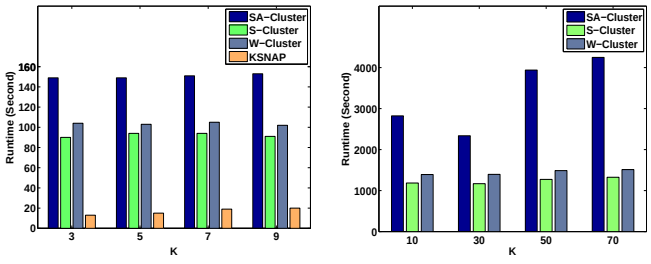


Figure 8: Weight Updates on DBLP



(a) Political Blogs (b) DBLP

Figure 9: Clustering Efficiency

contains authors who have close coauthor relationships, it also contains authors who work on the same topic but never collaborate. For example, both *Jiawei Han* and *Mohammed Javeed Zaki* are experts on frequent pattern mining, but they have never collaborated. As a result, S-Cluster assigns these two authors into two different clusters, since they are not reachable from each other based on random walks on coauthor relationship only. On the other hand, SA-Cluster can correctly assign these two researchers into the same cluster because they are connected with the same topic attribute. Similar cases can be found in other clusters as well, for example, *Rajeev Motwani* and *Nick Koudas*, *David J. DeWitt* and *Won Kim*, *Michael I. Jordan* and *Lise Getoor*, *James Allan* and *Justin Zobel*. They work on the same topics though they have never collaborated. S-Cluster fails to put any of these author pairs into the same cluster due to the lack of coauthor connectivity.

6. RELATED WORK

In literature, many graph clustering techniques have been proposed which mainly focused on the topological structures based on various criteria including normalized cut [19], modularity [16] or structural density [26]. The clustering results usually contain densely connected components within clusters. However, such methods largely ignore vertex attributes in the clustering process. On the other hand, Tian et al. [22] proposed OLAP-style aggregation approaches to summarize large graphs by grouping nodes based on user-selected attributes and relationships. Vertices in one group share the same attribute values and relate to vertices in another group through the same type of relationship. This method achieves homogeneous attribute values within clusters, but ignores the intra-cluster topological structures. As shown in our experiments, the generated partitions tend to have very low connectivity.

Other recent studies on graph clustering include the following. Sun et al. [20] proposed GraphScope which is able to discover communities in large and dynamic graphs, as well as to detect the changing time of communities. Sun et al. [21] proposed an algorithm, RankClus, which integrates clustering with ranking in large-scale information network analysis. The final results contain a set of clusters with a ranking of objects within each cluster. Navlakha et al. [15] proposed a graph summarization method using the MDL principle. Tsai and Chiu [25] developed a feature weight self-adjustment mechanism for K-Means clustering on relational datasets. In that study, finding feature weights is modeled as an optimization problem to simultaneously minimize the separations within clusters and maximize the separations between clusters. The adjustment margin of a feature weight is estimated by the importance of the feature in clustering. [4] proposed an algo-

Cluster 0 (DB)	Cluster 1 (DB)	Cluster 2 (DM)	Cluster 3 (ML)	Cluster 4 (IR)
streams 0.1247	database 0.2183	mining 0.2439	models 0.1894	retrieval 0.3066
over 0.0961	object 0.1738	patterns 0.1266	bayesian 0.1411	information 0.1176
k 0.0554	oriented 0.1229	frequent 0.1181	probabilistic 0.0964	document 0.0928
queries 0.0539	server 0.0526	sequential 0.0389	markov 0.0756	relevance 0.0538
nearest 0.0482	sql 0.0459	closed 0.0348	inference 0.0615	feedback 0.0369
neighbor 0.0430	dbms 0.0402	itemsets 0.0277	process 0.0446	expansion 0.0258
top 0.0402	relational 0.0231	maximal 0.0145	hidden 0.0438	ir 0.0249
approximate 0.0328	microsoft 0.0215	datasets 0.0100	latent 0.0404	cross 0.0230
computation 0.0309	management 0.0193	correlated 0.0082	mixture 0.0397	collection 0.0190
continuous 0.0267	panel 0.0176	efficiently 0.0072	gaussian 0.0307	evaluation 0.0138
skyline 0.0238	client 0.0156	sequences 0.0066	variable 0.0202	test 0.01322
aggregate 0.0216	design 0.0145	usage 0.0054	dirichlet 0.0189	translation 0.0119
efficient 0.0194	tuning 0.0141	periodic 0.0052	topic 0.0078	effectiveness 0.0107
sliding 0.0130	persistent 0.0098	evolving 0.0048	networks 0.0060	precision 0.0096
monitoring 0.0108	db 0.0093	pushing 0.0046	variational 0.0052	japanese 0.0094
uncertain 0.0107	cad 0.0084	subgraphs 0.0045	priors 0.0048	terms 0.0078
carlo 0.0105	directions 0.0056	projected 0.0044	discrete 0.0044	pseudo 0.0077

Table 4: Primary Topic of Each Cluster

rithm for mining communities on heterogeneous social networks. A method was designed for learning an optimal linear combination of different relations to meet users' expectation.

The concept of random walk has been widely used to measure vertex distances and similarities. Jeh and Widom [12] designed a measure called SimRank, which defines the similarity between two vertices in a graph by their neighborhood similarity. Pons and Latapy [18] proposed to use short random walks of length l to measure the similarity between two vertices in a graph for community detection. Other studies which use random walk with restarts include connection subgraph discovery [7] and center-piece subgraph discovery [23]. Liu et al. [14] proposed to use random walk with restart to discover subgraphs that exhibit significant changes in evolving networks.

7. CONCLUSIONS

In this paper, we solve the problem of graph clustering based on structural and attribute similarities. A unified neighborhood random walk distance measure is designed to measure vertex closeness on an attribute augmented graph. Based on this distance measure, we take a K-Medoids clustering approach to partition the graph into k clusters which have both cohesive intra-cluster structures and homogeneous attribute values. We provide theoretical analysis to quantitatively estimate the contributions of attribute similarity in the random walk distance measure. We further design a learning algorithm to adjust the degree of contributions of different attributes in the random walk model as we iteratively refine the clusters, and prove that the weights are adjusted towards the direction of clustering convergence. Experimental results on two real graphs demonstrate that our method achieves a very good balance between structural and attribute similarities.

8. ACKNOWLEDGMENTS

The work was supported in part by grants of the Research Grants Council of the Hong Kong SAR, China No. 419008 and the Chinese University of Hong Kong Direct Grant No. 2050446. We thank Yizhou Sun for providing us the DBLP data with topic information. We also thank Zheng Liu for his valuable comments.

9. APPENDIX: PROOF OF THEOREM 1

Proof. Firstly, we will prove that $P_V^l(v_j, v_p) = P_V^l(v_j, v_q), \forall v_j \in V$ and an arbitrary length l by induction. Since v_p and v_q have the same connectivity in G, we know that when $l = 1$,

$$P_V^1(v_j, v_p) = P_V^1(v_j, v_q),\ \forall v_j \in V$$

Assume that when $l = k$,

$$P_V^k(v_j, v_p) = P_V^k(v_j, v_q),\ \forall v_j \in V$$

Then when $l = k + 1$, $P_V^{k+1}(v_j, v_p) = \sum_{c=1}^{|V|} P_V^k(v_j, v_c) \cdot P_V(v_c, v_p)$, and $P_V^{k+1}(v_j, v_q) = \sum_{c=1}^{|V|} P_V^k(v_j, v_c) \cdot P_V(v_c, v_q)$. Since $P_V(v_c, v_p) = P_V(v_c, v_q)$, $\forall v_c \in V$, we have

$$P_V^{k+1}(v_j, v_p) = P_V^{k+1}(v_j, v_q),\ \forall v_j \in V \tag{28}$$

Secondly, we will prove that $C^l(v_i, v_p) > C^l(v_i, v_q)$, for $v_i \in V$ which shares more attribute values with v_p than with v_q, and an arbitrary length l by induction. According to Eq.(12), we have

$$C(v_i, v_p) > C(v_i, v_q)$$

and

$$C(v_j, v_p) = C(v_j, v_q), \forall v_j \neq v_i, v_p, v_q$$

Assume that when $l = k$,

$$C^k(v_i, v_p) > C^k(v_i, v_q)$$

Then when $l = k + 1$, $C^{k+1}(v_i, v_p) = \sum_{c=1}^{|V|} C^k(v_i, v_c) \cdot C(v_c, v_p)$, and $C^{k+1}(v_i, v_q) = \sum_{c=1}^{|V|} C^k(v_i, v_c) \cdot C(v_c, v_q)$. $\forall v_c \neq v_i, v_p, v_q$, we have $C(v_c, v_p) = C(v_c, v_q)$. Then we only need to prove that

$$\sum_{v_c = v_i, v_p, v_q} C^k(v_i, v_c) \cdot C(v_c, v_p) > \sum_{v_c = v_i, v_p, v_q} C^k(v_i, v_c) \cdot C(v_c, v_q)$$

or equivalently

$$\sum_{v_c = v_i, v_p, v_q} C^k(v_i, v_c) \cdot C(v_c, v_p) - \sum_{v_c = v_i, v_p, v_q} C^k(v_i, v_c) \cdot C(v_c, v_q) > 0 \tag{29}$$

Since we have $C(v_p, v_p) = C(v_q, v_q) \geq C(v_q, v_p)$, we then have

$$C^k(v_i, v_i) \cdot (C(v_i, v_p) - C(v_i, v_q)) > 0 \tag{30}$$

$$(C^k(v_i, v_p) - C^k(v_i, v_q)) \cdot (C(v_p, v_p) - C(v_q, v_p)) \geq 0 \tag{31}$$

Combining Eqs.(30) and (31), we can prove Eq.(29). In addition, $\forall v_j \neq v_i, v_p, v_q$, $C^l(v_j, v_p) = C^l(v_j, v_q)$ for an arbitrary l. Therefore, we have

$$C^l(v_c, v_p) \geq C^l(v_c, v_q), \forall v_c \neq v_p, v_q \tag{32}$$

Next, according to Lemma 1, T_l can be represented in the form of $\sum \prod_{m_i \geq 0, n_i \geq 0} (P_V^{m_i} C^{n_i})$. We will prove that

$$(P_V^{m_i} C^{n_i})(v_i, v_p) > (P_V^{m_i} C^{n_i})(v_i, v_q) \tag{33}$$

We have

$$(P_V^{m_i} C^{n_i})(v_i, v_p) = \sum_{c=1}^{|V|} P_V^{m_i}(v_i, v_c) \cdot C^{n_i}(v_c, v_p)$$

and

$$(P_V^{m_i} C^{n_i})(v_i, v_q) = \sum_{c=1}^{|V|} P_V^{m_i}(v_i, v_c) \cdot C^{n_i}(v_c, v_q)$$

Since we have proven Eqs.(28) and (32), we can prove Eq.(33), which then means that $T_l(v_i, v_p) > T_l(v_i, v_q)$. According to the relation between P_A^l and T_l, $P_A^l(v_i, v_j) = T_l(v_i, v_j), v_i, v_j \in V$, we have the following conclusion:

$$P_A^l(v_i, v_p) > P_A^l(v_i, v_q)$$

10. REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 94–105, Seattle, WA, June 1998.

[2] T. M. Apostol. *Calculus, Vol. 1: One-Variable Calculus, with an Introduction to Linear Algebra, 2nd edition*. Wiley, 1967.

[3] L. Botton and Y. Bengio. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems 7 (NIPS'94)*, pages 585–592, Denver, CO, Dec. 1994.

[4] D. Cai, Z. Shao, X. He, X. Yan, and J. Han. Mining hidden community in heterogeneous social networks. In *Proc. Workshop on Link Discovery: Issues, Approaches and Applications (LinkKDD'05)*, pages 58–65, Chicago, IL, Aug. 2005.

[5] R. Descartes. *The Geometry of René Descartes*. Dover Publications, 1954.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. In *Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining (KDD'96)*, pages 226–231, Portland, OR, Aug. 1996.

[7] C. Faloutsos, K. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *Proc. 2004 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'04)*, pages 118–127, Seattle, WA, Aug. 2004.

[8] B. Fine and G. Rosenber. *The Fundamental Theorem of Algebra (Undergraduate Texts in Mathematics)*. Springer-Verlag, 1997.

[9] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *Proc. 9th ACM Conf. Hypertext and Hypermedia*, pages 225–234, Pittsburgh, PA, June 1998.

[10] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, pages 58–65, New York, NY, Aug. 1998.

[11] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. 1999 Int. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR'99)*, pages 50–57, Berkeley, CA, Aug. 1998.

[12] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *Proc. 2002 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'02)*, pages 538–543, Edmonton, Canada, July 2002.

[13] L. Kaufman and P. J. Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis based on the L1 Norm*, pages 405–416, 1987.

[14] Z. Liu, J. X. Yu, Y. Ke, X. Lin, and L. Chen. Spotting significant changing subgraphs in evolving graphs. In *Proc. 2008 Int. Conf. Data Mining (ICDM'08)*, Pisa, Italy, Dec. 2008.

[15] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *Proc. 2008 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'08)*, pages 419–432, Vancouver, Canada, June 2008.

[16] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. In *Phys. Rev. E 69, 026113*, 2004.

[17] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 144–155, Santiago, Chile, Sept. 1994.

[18] P. Pons and M. Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10(2):191–218, 2006.

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 22, no. 8, pages 888–905, 2000.

[20] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proc. 2007 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'07)*, pages 687–696, San Jose, CA, 2007.

[21] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu. Rankclus: Integrating clustering with ranking for heterogenous information network analysis. In *Proc. 2009 Int. Conf. Extending Database Technology (EDBT'09)*, pages 565–576, Saint Petersburg, Russia, Mar. 2009.

[22] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *Proc. 2008 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'08)*, pages 567–580, Vancouver, Canada, June 2008.

[23] H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In *Proc. 2006 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'06)*, pages 404–413, Philadelphia, PA, 2006.

[24] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *Proc. 2006 Int. Conf. on Data Mining (ICDM'06)*, pages 613–622, Hong Kong, Dec. 2006.

[25] C.-Y. Tsai and C.-C. Chui. Developing a feature weight self-adjustment mechanism for a k-means clustering algorithm. *Computational Statistics and Data Analysis*, 52:4658–4672, 2008.

[26] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. Scan: a structural clustering algorithm for networks. In *Proc. 2007 Int. Conf. Knowledge Discovery and Data Mining (KDD'07)*, pages 824–833, San Jose, CA, Aug. 2007.

[27] C. Zhai, A. Velivelli, and B. Yu. A cross-collection mixture model for comparative text mining. In *Proc. 2004 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'04)*, pages 743–748, Seattle, WA, Aug. 2004.