Clustering Large Attributed Graphs: A Balance between Structural and Attribute Similarities

HONG CHENG, YANG ZHOU, and JEFFREY XU YU, The Chinese University of Hong Kong

Social networks, sensor networks, biological networks, and many other information networks can be modeled as a large graph. Graph vertices represent entities, and graph edges represent their relationships or interactions. In many large graphs, there is usually one or more attributes associated with every graph vertex to describe its properties. In many application domains, graph clustering techniques are very useful for detecting densely connected groups in a large graph as well as for understanding and visualizing a large graph. 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. 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 article, we propose a novel graph clustering algorithm, *SA-Cluster*, which achieves a good balance between 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 quickly through iterative cluster refinement. Some optimization techniques on matrix computation are proposed to further improve the efficiency of SA-Cluster on large graphs. Extensive experimental results demonstrate the effectiveness of SA-Cluster through comparisons with the state-of-the-art graph clustering and summarization methods.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Graph clustering, structural proximity, attribute similarity

ACM Reference Format:
Cheng, H., Zhou, Y., and Yu, J. X. 2011. Clustering large attributed graphs: A balance between structural and attribute similarities. ACM Trans. Knowl. Discov. Data 5, 2, Article 12 (February 2011), 33 pages.
DOI = 10.1145/1921632.1921638 http://doi.acm.org/10.1145/1921632.1921638

1. INTRODUCTION

Clustering is a useful and important unsupervised learning technique widely studied in literature [Ng and Han 1994; Ester et al. 1996; Agrawal et al. 1998; Gibson et al. 1998]. The general goal of clustering is to group similar objects into one cluster while

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 Grants No. 2050446 and No. 2050473.
Authors' address: H. Cheng, Y. Zhou, and J. X. Yu, Department of Systems Engineering and Engineering Management, William M. W. Mong Engineering Building, Room 609, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong; email: {hcheng, zhouy, yu}@se.cuhk.edu.hk.
Permission to make digital or hard copies part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from the Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2011 ACM 1556-4681/2011/02-ART12 $10.00
DOI 10.1145/1921632.1921638 http://doi.acm.org/10.1145/1921632.1921638

partitioning dissimilar objects into different clusters. Clustering has broad applications in the analysis of business and financial data, biological data, time series data, spatial data, trajectory data, and so on.

Many information networks are nowadays available for analysis, including social networks, sensor networks, and biological networks. As a particular example, social networks are expanding quickly with the emergence and rapid proliferation of social applications and media. Graph as an expressive data structure is popularly used to model structural relationship between objects in many application domains mentioned above. Graph clustering is an interesting and challenging research problem which has received much attention recently [Shi and Malik 2000; Newman and Girvan 2004; Xu et al. 2007]. Clustering on a large graph aims to partition the graph into several densely connected components. This is very useful for understanding and visualizing large graphs. 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 cuts [Shi and Malik 2000], modularity [Newman and Girvan 2004] or structural density [Xu et al. 2007]. On the other hand, one recent graph summarization method [Tian et al. 2008] aims to partition a 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) or structural similarity (e.g., the number of common neighbors of two vertices) while relational data clustering measures distance between two data points mainly based on attribute similarity (e.g., Euclidean distance between two attribute vectors).

In the information networks formed from many real applications, both the graph topological structure and the vertex properties are important features for analysis. For example, in a social network, vertex properties describe roles of a person while the topological structure represents relationships among a group of people. To partition a large graph into clusters, it is natural to assign those vertices which are closely connected and share similar characteristics into the same cluster. However, existing 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 intracluster structure. An ideal graph clustering should generate clusters which have a *cohesive* intracluster 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 two clusters in several possible ways depending on the clustering criteria.

— *Structure-based clustering*. Figure 1(b) shows a clustering result based on vertex connectivity, that is, coauthor relationship. Authors within clusters are closely connected. However, in one of the clusters the authors have quite different topics; for example, half of them work on XML and the other half work on *Skyline*.

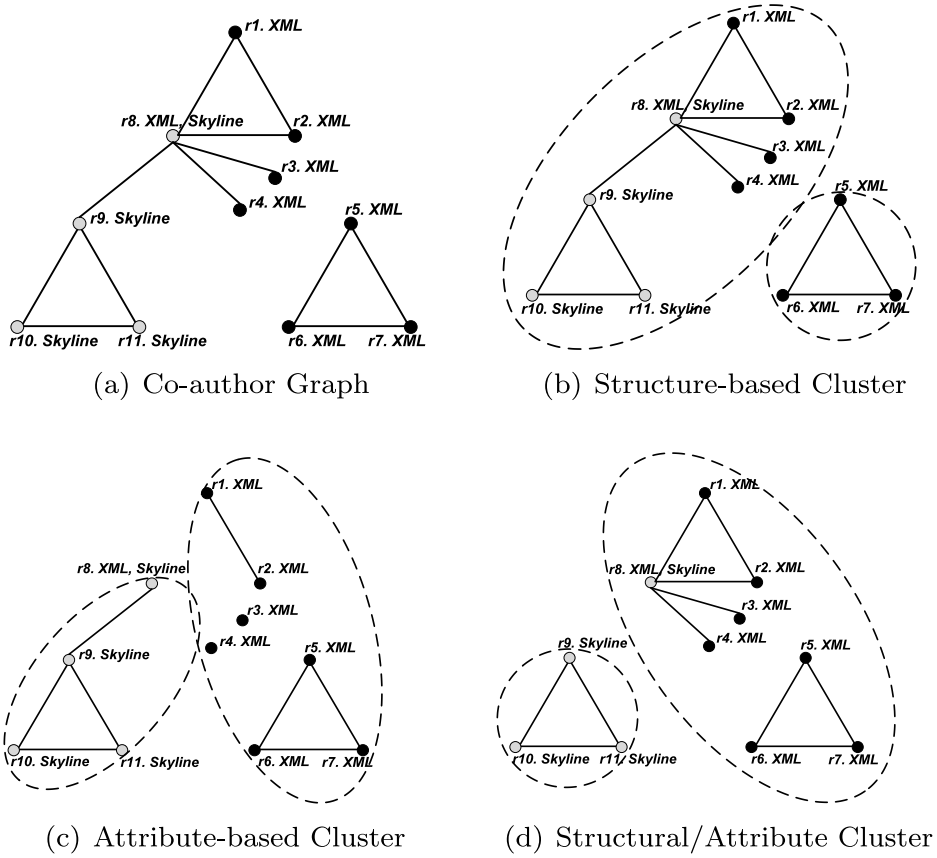


Fig. 1. A coauthor network example with an attribute "topic."

—*Attribute-based clustering*. Figure 1(c) shows another clustering result based on attribute similarity, that is, topics. Authors within clusters work on the same topics. However, the coauthor relationship is lost due to the partitioning so that authors are quite isolated in one of the clusters.
—*Structure and attribute-based 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. This is the result we want to achieve in this work.

The problem we study in this article is to cluster a large-scale graph associated with vertex attributes based on both structural and attribute similarities. We call such a graph with vertex attributes an *attributed graph*. The goal is to partition the attributed 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, affiliation 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 article, 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, the original vertices are called *structure vertices* and the original edges are called *structure edges*. With such graph augmentation, we express the attribute similarity as vertex vicinity in the graph: two vertices that 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 article, 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 are of different types and 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 changed. 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?
(3) *Fast Random Walk Distance Computation.* To use the neighborhood random walk model, we have to perform matrix multiplication on a transition probability matrix of the attributed graph to calculate the pairwise vertex closeness values. As matrix multiplication is expensive, we need to design some optimization techniques on matrix operation to improve the efficiency.

We will address the challenges we have listed and propose our graph clustering algorithm based on a unified neighborhood random walk distance. A preliminary study

on this problem appeared in Zhou et al. [2009]. The main contributions of this article are summarized in the following.

(1) We study the problem of clustering large 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 that share attribute values. As a result, vertices become closer if they share attribute values. A neighborhood random walk model is used to measure the vertex closeness on the augmented graph through structure 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 attribute edge weights are adjusted towards the direction of clustering convergence.
(4) We also design some techniques based on the Matrix Neumann Series [Strang 2005] to speed up the matrix computation for random walk distance calculation. It reduces the number of matrix multiplication from $O(l)$ to $O(log_2 l)$ where l is the length limit of the random walks.
(5) We perform extensive evaluation of our proposed clustering approach by using large-scale real social 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 article 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 introduces the optimization techniques on matrix computation based on the Matrix Neumann Series for efficiency improvement. Section 6 presents extensive experimental results. Some discussions are presented in Section 7, followed by related work on graph clustering and graph mining in Section 8. Finally, Section 9 concludes the article.

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 it is easy to reach v_j from v_i and vice versa. In this sense, we say v_i and v_j 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

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

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

$$R^l = \sum_{\gamma=0}^{l} c(1-c)^{\gamma} P^{\gamma}. \tag{3}$$

Here, P is the transition probability matrix for graph G, and R is the neighborhood random walk distance matrix. $c(1-c)^{\gamma}$ implies the probability of jumping back to the initial starting vertex after a random walk of γ steps without restart. According to Equation (3), the recursive form of the random walk distance matrix is

$$R^l = \sum_{\gamma=0}^{l} c(1-c)^{\gamma} P^{\gamma} = c(1-c)^l P^l + R^{l-1}. \tag{4}$$

Then the structural closeness between two vertices v_i and v_j is

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

Note that in the neighborhood random walk, we only consider random walk within length l, that is, random walks are only conducted in the l-step neighborhood of a vertex with a restart probability c. When $l \to \infty$, the neighborhood random walk is the same as the random walk with restart defined in existing work, [Tong et al. 2006].

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 use a weighted distance function as in Equation (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

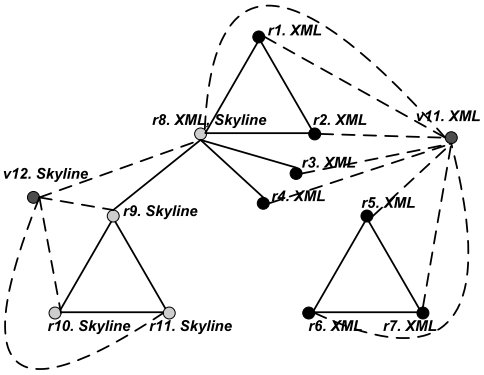


Fig. 2. Attribute augmented graph with topics.

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 $|Dom(a_i)| = n_i$ is the size of the domain for a_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, for example, authors 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 neighbor, that is, 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

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

Fig. 3. Transition probability matrix of the attribute augmented graph example.

to a_1, a_2, ..., a_m have an edge weight of ω_1, ω_2, ..., ω_m, respectively. Therefore, the transition probability from vertex v_i to vertex v_j through a structure edge is

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

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

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

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

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

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

$$p_{v_{ip},v_{jq}} = 0, \forall v_{ip}, v_{jq} \in V_a. \tag{8}$$

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

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

Combining Equations (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 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:

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

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 on an attribute augmented graph is

$$R_A^l = \sum_{\gamma=0}^{l} c(1-c)^{\gamma} P_A^{\gamma}. \tag{11}$$

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 the original graph G.

3.3 Some Analysis on the Unified Distance

In this part, we will provide some theoretical analysis to show how the attribute similarity can increase vertex closeness in the unified random walk distance measure. 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 Equation (5); $A = [A_1, A_2, \ldots, A_N]^T$ is a $|V| \times |V_a|$ matrix representing the transition probabilities defined by Equation (6); $B = [B_1, B_2, \ldots, B_N]$ 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

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

v_{kp} is the value taken by vertex v_i on attribute a_k, $k = 1, \ldots, m$. 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 in Λ.

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 $l = k + 1$,

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

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

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

□

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, that is, $\forall v_j \in V$, $(v_j, v_p) \in E$, 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, that is, $\forall v_j \in V$, $(v_j, v_p) \in E$, 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 $P_A^{\gamma}(v_i, v_p) > P_A^{\gamma}(v_i, v_q)$ for an arbitrary γ. According to Equation (11), R_A^l is a linear sum of P_A^{γ} with positive coefficients $c(1-c)^{\gamma}$, $\gamma = 0, \ldots, l$. 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 $\omega_0 \neq \omega_1 \neq \ldots \neq \omega_m$. 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 $\omega_1, \ldots, \omega_m$?

Many existing graph clustering/partitioning methods focused only on either topological structures [Shi and Malik 2000; Newman and Girvan 2004; Xu et al. 2007] or vertex attributes [Tian et al. 2008]. Many of them cannot be directly applied to cluster the attribute augmented graph, because they usually work on a homogeneous graph, that is, a graph with the same type of nodes and the same type of edges. On the attribute augmented graph which is heterogeneous, they lack a mechanism to differentiate and learn the different contributions of the structure edges and attribute edges. As a result, they lack a mechanism to balance the structural and attribute similarities.

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, that is, there exist very few or no neighborhood random walk paths 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 [Kaufman and Rousseeuw 1987]: 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 $\omega_1, \ldots, \omega_m$

[1]We fix $\omega_0 = 1.0$ and adjust $\omega_1, \ldots, \omega_m$ relative to ω_0.
[2]Different from traditional distance measures, a random walk distance measures the closeness between two vertices.

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 [Hinneburg and Keim 1998]. 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. On the other hand, if the l-step neighborhood of a vertex v_i is sparse, then v_i may not be a good candidate as a cluster centroid. 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 top-k vertices from the sorted list as the initial centroids $\{c_1^0, \ldots, c_k^0\}$.

4.2 Clustering Process

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

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

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

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

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 intracluster similarity and minimize intercluster 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 intracluster 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 $W = \{\omega_1, \ldots, \omega_m\}$ on attribute edges, a random walk length limit l, and the cluster number k, the goal of 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 $\omega_i \geqslant 0$, $i = 1, \ldots, m$.

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 now study the relationship between the weights $W = \{\omega_1, \ldots, \omega_m\}$ and the objective function in Equation (18).

THEOREM 3. *Given an arbitrary partition $\{V_i^*\}_{i=1}^k$ of graph G, there exists a unique solution $W^* = \{\omega_1^*, \ldots, \omega_m^*\}$ s.t. $\sum_{i=1}^m \omega_i^* = m$, $\omega_i^* \geqslant 0$, $i = 1, \ldots, m$, which maximizes the objective function in Equation (18).*

PROOF. According to Equation (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 multivariable $\omega_1, \ldots, \omega_m$ with nonnegative real coefficients. In addition, the coefficients $c(1-c)^\gamma$ in Equation (11) are positive. Then we can infer that $R_A^l(v_p, v_q)$ is a polynomial function of $\omega_1, \ldots, \omega_m$ with nonnegative 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 $\omega_1, \ldots, \omega_m$ with nonnegative 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 to 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 Equation (20). $g_i(\omega_i)$ is a polynomial function of ω_i with non-negative real coefficients.

Gauss's Fundamental Theorem of Algebra [Fine and Rosenber 1997] states that every non-constant single-variable polynomial with complex coefficients has at least one complex root. Thus, Equation (20) must have at least one complex root. For $\lambda = \lambda^*$, there are two possible cases, that is, $\lambda^* \geq -c$ and $\lambda^* < -c$, to be discussed separately as follows. □

Case 1. $\lambda^* \geq -c$. According to Descartes's Rule of Signs [Descartes 1954], 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 nonnegative real numbers. Thus the number of positive roots of Equation (20) is 0 according to Descartes's Rule of Signs. Actually we have

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

for any assignments $\omega_i \geq 0$, $i = 1, \ldots, m$. 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 Equation (20). According to Descartes's Rule of Signs, there exists one or no positive root in Equation (20) because there is one sign difference in the equation.

Next, we will prove that there exists at least one solution to Equation (20) if $\lambda^* < -c$. When $\omega_i = 0$, $i = 1, \ldots, m$, 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 nonnegative, 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 [Apostol 1967], we can conclude that there exists at least one solution to Equation (20) in the open interval $(0, \delta)$.

To summarize, for some $\lambda^* < -c$, there exists a unique positive root $W^* = \{\omega_1^*, \ldots, \omega_m^*\}$ for Equation (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 $\omega_1, \ldots, \omega_m$ 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 that determines whether two vertices share an attribute value.

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

Then $\triangle\omega_i^t$ is estimated by counting the number of vertices within clusters that share attribute values with the centroids on a_i. The larger number of vertices that 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 Equation (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 $\omega_i^{t+1} > \omega_i^t$, that is, 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. Here we 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, that is, $\omega_i^{t+1} > \omega_i^t$; on the other hand, if vertices within clusters have quite different attribute

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 $V_1, \ldots, V_k$.

1: Initialize $\omega_1 = \ldots = \omega_m = 1.0$, 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^* where the centroid $c^* = \arg\max_{c_j} d(v_i, c_j)$;
6: Update cluster centroids with the most centrally located point in each cluster according to Eqs. (15)–(16);
7: Update weights $\omega_1, \ldots, \omega_m$ according to Eq. (25);
8: Re-calculate R_A^l;
9: Return k clusters $V_1, \ldots, V_k$.

values on a_i, the weight is then decreased, that is, $\omega_i^{t+1} < \omega_i^t$. 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 intracluster random walk distances. This is analogous to the *The rich get richer and the poor get poorer* principle, as attributes with a very good clustering tendency will gain more weights for an increasing influence in the random walk distance, while attributes with very random values will have decreasing weights. We have proven in Theorem 3 that there exists a unique solution $W^* = \{\omega_1^*, \ldots, \omega_m^*\}$ which maximizes the objective function, thus we can conclude that the weight self-adjustment strategy increases the objective function towards convergence.

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 SKETCH. *Existing work has studied the convergence properties of the partitioning clustering algorithms, such as K-Means [Botton and Bengio 1994]. 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 6, we will demonstrate through experiments that our clustering algorithm converges very quickly on real datasets.

5. EFFICIENT COMPUTATION OF RANDOM WALK DISTANCE

5.1 Reducing the Number of Matrix Multiplications

One issue with SA-Cluster is the computational complexity. We need to compute N^2 pairs of random walk distances between vertices in V through matrix multiplication.

As $W = \{\omega_1, \ldots, \omega_n\}$ is updated, the random walk distances need to be recalculated, as shown in lines 7-8 in Algorithm 1. The cost analysis of SA-Cluster can be expressed as follows.

$$t \cdot (T_{random\_walk} + T_{centroid\_update} + T_{assignment}), \tag{26}$$

where t is the number of iterations in the clustering process, T_{random_walk} is the cost of computing the random walk distance matrix R_A, $T_{centroid\_update}$ is the cost of updating cluster centroids, and $T_{assignment}$ is the cost of assigning all points to cluster centroids.

The time complexity of $T_{centroid\_update}$ and $T_{assignment}$ is $O(n)$, since each of these two operations performs a linear scan of the graph vertices. On the other hand, the time complexity of T_{random_walk} is $O(n^3)$ because the random walk distance calculation consists of a series of matrix multiplication and addition. According to Equation (11), $R_A^l = \sum_{\gamma=0}^{l} c(1-c)^\gamma P_A^\gamma$ where l is the length limit of a random walk. To compute R_A^l, we have to compute $P_A^2, P_A^3, \ldots, P_A^l$, that is, $(l-1)$ matrix multiplication operations in total. It is clear that T_{random_walk} is the dominant factor in the clustering process.

In this section, we aim to improve the efficiency of SA-Cluster with a fast matrix computation technique. In other words, we are studying how to compute the random walk distance matrix R_A^l more efficiently. The core idea is to use the Matrix Neumann Series [Strang 2005] to reduce the number of matrix multiplication operations.

We first introduce some preliminary concepts before discussing the fast random walk computation. Given a square matrix A, $\sum_{i=0}^{\infty} c_i A^i$ is called the *Power Series* of A (assume $A^0 = I$). In particular, when all coefficients c_i $(i = 0, 1, \ldots)$ are equal to 1, the Power series of A is reduced to $\sum_{i=0}^{\infty} A^i$. We call it the *Neumann Series* of A. In related literature, we can find the following theorems on Neumann series. (Theorems 5 to 9 and the detailed proofs can be found in Strang [2005] and Manning et al. [2008].)

THEOREM 5. *The Neumann series of A converges in the normed space X if and only if A is a convergent matrix, that is,* $\lim_{i\to\infty} A^i = 0$.

THEOREM 6. *An arbitrary square matrix A is convergent if and only if the module of each eigenvalue of A is smaller than 1.*

THEOREM 7. *If the infinite Neumann series of a square matrix A converges in the normed space X, then $I - A$ is invertible and its inverse is the sum of the series:*

$$(I - A)^{-1} = \sum_{i=0}^{\infty} A^i, \tag{27}$$

where I is the identity matrix in X.

The Neumann series provides approximations of $(I - A)^{-1}$ when A has entries of small magnitude. For example, a first-order approximation is $(I - A)^{-1} \approx I + A$.

We know that a square matrix is invertible if and only if its determinant is not equal to 0 or it has a full rank. A matrix that is invertible is called nonsingular matrix. Singular matrices are rare in the sense that if we pick a random square matrix, it is almost surely not singular.

THEOREM 8. *If $I - A$ is invertible, then the sum of the finite Neumann series of a square matrix A is as follow:*

$$\sum_{i=0}^{k} A^i = (I - A)^{-1} \cdot (I - A^{k+1}), \tag{28}$$

where I is the identity matrix in X.

If we compare the right-hand side of Equation (11), that is, $\sum_{\gamma=0}^{l} c(1-c)^{\gamma} P_A^{\gamma}$ with the left-hand side of Equation (28), that is, $\sum_{i=0}^{k} A^i$, we find that they are very much alike. Equation (11) is actually the sum of the finite power series of the transition probability matrix P_A. If we denote $(1-c)P_A$ as a new square matrix P' and move the factor c outside of $\sum$, then R_A^l is the sum of the finite Neumann series of P'. Based on Equation(28), we can rewrite the random walk distance matrix as

$$R_A^l = \sum_{\gamma=0}^{l} c(1-c)^{\gamma} P_A^{\gamma} = c(I - (1-c)P_A)^{-1} \cdot (I - ((1-c)P_A)^{l+1}). \tag{29}$$

To compute $((1-c)P_A)^{l+1}$ in Equation (29), we need to compute $((1-c)P_A)^2$, $((1-c)P_A)^4$, ..., $((1-c)P_A)^{2^{\lfloor \log(l+1) \rfloor}}$. Thus the number of matrix multiplication operations can be reduced from $(l-1)$ to $(\lceil \log_2(l+1) \rceil + 1)$. The additional "1" here refers to the computation for $(I - (1-c)P_A)^{-1}$.

We call our clustering algorithm which uses Equation (29) for random walk distance computation *SA-Cluster-Opt*, and call the original version which uses Equation (11) *SA-Cluster*. These two algorithms differ only in the number of matrix multiplication operations, but achieve the same clustering results. We will compare their efficiency in Section 6. We will also compare the efficiency of random walk computation by our technique based on Neumann series with the state-of-the-art computation technique for fast random walk with restart proposed by Tong et al. [2006].

5.2 Handling The Noninvertible Matrix

In the above computation defined in Equation (29), we assume that the matrix $I - (1-c)P_A$ is invertible. We denote the matrix $C = I - (1-c)P_A$. If it is not invertible, the Neumann series formula cannot be directly applied to solve the problem. In this section, we will propose techniques to address this problem. The core idea is to attempt to find an invertible matrix C_n to approximate the noninvertible matrix $C = I - (1-c)P_A$ while minimizing the difference between the two matrices.

The first step is to construct the Singular Value Decomposition (SVD) form of the matrix $C = I - (1-c)P_A$. There is a theorem on SVD in Manning et al. [2008].

THEOREM 9. *Suppose r is the rank of an m-by-n matrix C whose entries come from the field K, which is either the field of real numbers or the field of complex numbers. Then there exists a singular value decomposition of the form:*

$$C = U \Sigma V^T \tag{30}$$

where U is an m-by-m unitary matrix over K and the columns of U are the orthogonal eigenvectors of CC^T; Σ is an m-by-n diagonal matrix with nonnegative real numbers on the diagonal; and V is an n-by-n unitary matrix over K and the columns of V are the orthogonal eigenvectors of C^TC.

Suppose $\lambda_1, \ldots, \lambda_r$ are the eigenvalues of C^TC or CC^T, with $\lambda_i \geq \lambda_{i+1}$. Let $\sigma_i = \sqrt{\lambda_i}$ for $1 \leq i \leq r$. Then we have $\Sigma_{ii} = \sigma_i$ in the matrix Σ for $1 \leq i \leq r$, and zero otherwise. The diagonal entries Σ_{ii} are ordered in the descending order. In this case, the diagonal matrix Σ is uniquely determined by C but not the matrices U or V. The diagonal entries Σ_{ii} are known as the singular values of C.

Next, we will use singular value decomposition to create an invertible matrix C_n to approximate the non-invertible matrix C. Since C_n has a full rank, we call this problem the *full-rank matrix approximation* problem. It is actually a reverse operation of the commonly used low-rank approximation operation. Given an n-by-n square matrix C with a rank r and an n-by-n square matrix C_n with a full rank n, the matrix difference between C and C_n is measured by the Frobenius norm of $X = C_n - C$. The Frobenius norm can be defined as:

$$\|X\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}^2}. \tag{31}$$

Our goal is to find a new matrix C_n that minimizes the Frobenius norm, while constraining C_n with the full rank n. The following three steps are adopted to solve this full-rank approximation problem:

(1) Compute the SVD form of an n-by-n square matrix $C = U\Sigma V^T$.
(2) A new matrix Σ' is derived by substituting the $n-r$ zero diagonal entries of Σ with different and very small nonzero values.
(3) Calculate the full-rank approximation of $C_n = U\Sigma' V^T$.

Theoretical analysis of this full-rank approximation solution is given in the following theorem.

THEOREM 10. *C_n is the full-rank approximation of C to minimize the Frobenius norm of $X = C_n - C$.*

PROOF. Since the Frobenius norm is unitarily invariant and the SVD form of C is $C = U\Sigma V^T$, an equivalent statement can be generated as follows:

$$\min_{C_n} \|U^T C_n V - \Sigma\|_F, \tag{32}$$

as $U^TU = V^TV = I$. According to step (3) listed above, we compute C_n by $U\Sigma' V^T$. Thus we have:

$$U^T C_n V = U^T(U\Sigma' V^T)V = \Sigma'.$$

Thus,

$$\min_{C_n} \|U^T C_n V - \Sigma\|_F = \min_{C_n} \|\Sigma' - \Sigma\|_F = \min_{\sigma_i'} \sqrt{\sum_{i=1}^{n}(\sigma_i' - \sigma_i)^2}, \tag{33}$$

Σ has r nonzero diagonal entries and Σ' has n nonzero diagonal entries. When $\sigma_i = \sigma_i'$ for $1 \leq i \leq r$, Equation (33) is equivalent to:

$$\min_{\sigma_i'} \sqrt{\sum_{i=1}^{r}(\sigma_i' - \sigma_i)^2 + \sum_{i=r+1}^{n} \sigma_i'^2} = \min_{\sigma_i'} \sqrt{\sum_{i=r+1}^{n} \sigma_i'^2}. \tag{34}$$

Here σ_i' for $1 \leq i \leq n$ are different from each other since both $C_n^T C_n$ and $C_n C_n^T$ are full rank matrices. To minimize Equation (34), σ_i' for $r+1 \leq i \leq n$ should be very small positive numbers. Thus, the previous equation is close to zero:

$$\min_{\sigma_i'} \sqrt{\sum_{i=r+1}^{n} \sigma_i'^2} \approx 0. \tag{35}$$

Therefore, C_n is the full-rank approximation of C to minimize the Frobenius norm when $\sigma_i' = \sigma_i$ for $1 \leq i \leq r$ and $\sigma_i' \approx 0$ for $r+1 \leq i \leq n$. □

Therefore, we can substitute a singular matrix $C = I - (1-c)P_A$ with a full-rank matrix C_n in the finite Neumann series in Equation (29).

In literature, pseudoinverse [Penrose 1956] is another technique to compute the matrix inverse of a noninvertible matrix. One way to get the pseudoinverse is by using the singular value decomposition. If we have $C = U\Sigma V^T$, then the pseudoinverse of C is $C^{-1} = V\Sigma^{-1}U^T$. For the diagonal matrix Σ, the pseudoinverse Σ^{-1} is computed by taking the reciprocal of each nonzero element on the diagonal and keeping the zero diagonal elements.

In Section 6, we will evaluate and compare the approximation quality of the random walk distance with this full-rank approximation technique and the pseudoinverse method, when the matrix C is not invertible.

6. EXPERIMENTAL STUDY

In this section, we performed extensive experiments to evaluate the performance of our algorithms SA-Cluster and SA-Cluster-Opt on real graph datasets. All experiments were done on a Dell PowerEdge R900 server with four 2.67GHz six-core CPUs and 128GB main memory running Windows Server 2008. All algorithms were implemented in Matlab.

6.1 Experimental Datasets

We use three 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 U.S. 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 5,000 Authors. We use the DBLP Bibliography data from four research areas of database (DB), data mining (DM), information retrieval (IR) and artificial intelligence (AI)[3]. 5,000 most prolific authors from the four areas are selected for clustering. We build a coauthor graph where nodes represent authors and edges represent their coauthor relationships. In addition, we use two relevant attributes: *prolific* and *primary topic* to describe an author. For the 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 the attribute "primary topic," we use a topic modeling approach [Hofmann 1998; Zhai et al. 2004] to extract 100 research topics from a document collection composed of paper titles from

[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.

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.

DBLP Dataset 84,170 Authors. To test the efficiency and scalability of our method, we use a larger DBLP coauthor network with 84,170 authors, selected from the following areas: database, data mining, information retrieval, machine learning, artificial intelligence, computer systems, theory, computer vision, architecture, programming language, networking, simulation, natural language processing, multimedia, and human-computer interaction. The coauthor graph and the two vertex attributes are defined similarly as in the 5,000 author network except the attribute "prolific." We study the distribution of the publication numbers from the 84,170 authors and find a more suitable partition on this larger dataset is: authors with ≥ 20 papers are labeled as highly prolific; authors with ≥ 5 and < 20 papers are labeled as prolific and authors with < 5 papers are labeled as low prolific.

6.2 Comparison Methods and Evaluation

(1) *SA-Cluster*. This is our proposed algorithm which considers both structural and attribute similarities. The improved version **SA-Cluster-Opt** based on Neumann series is also tested for efficiency evaluation.
(2) *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 $\omega_1 = \ldots = \omega_m = 0$.
(3) *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.
(4) *k-SNAP*. We implemented k-SNAP Top-Down [Tian et al. 2008] that groups vertices with the same attribute values into one cluster.

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.

$$density(\{V_i\}_{i=1}^k) = \sum_{i=1}^{k} \frac{|\{(v_p, v_q)|v_p, v_q \in V_i, (v_p, v_q) \in E\}|}{|E|} \tag{36}$$

$$entropy(\{V_i\}_{i=1}^k) = \sum_{i=1}^{m} \frac{\omega_i}{\sum_{p=1}^{m} \omega_p} \sum_{j=1}^{k} \frac{|V_j|}{|V|} entropy(a_i, V_j) \tag{37}$$

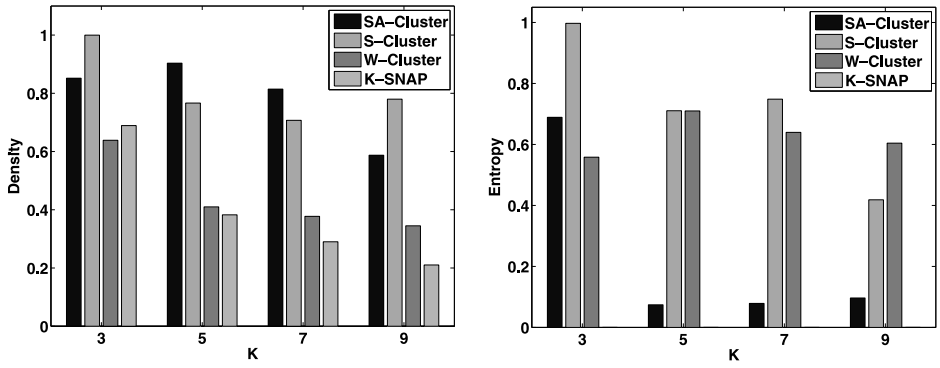
where

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

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.

6.3 Cluster Quality Evaluation

Figure 4(a) shows the density comparison among the four methods on Political Blogs when we set the cluster number $k = 3, 5, 7, 9$. The density values by SA-Cluster and



(a) density (b) entropy

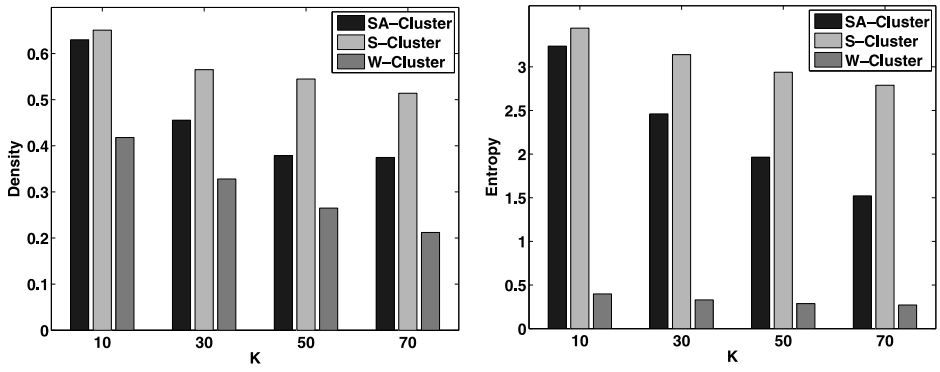
Fig. 4. Cluster quality comparison on political blogs.

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 among 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. SA-Cluster achieves a much lower entropy than S-Cluster. When $k = 5, 7$ or 9, the entropy by SA-Cluster is as low as less than 0.1 while the 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.

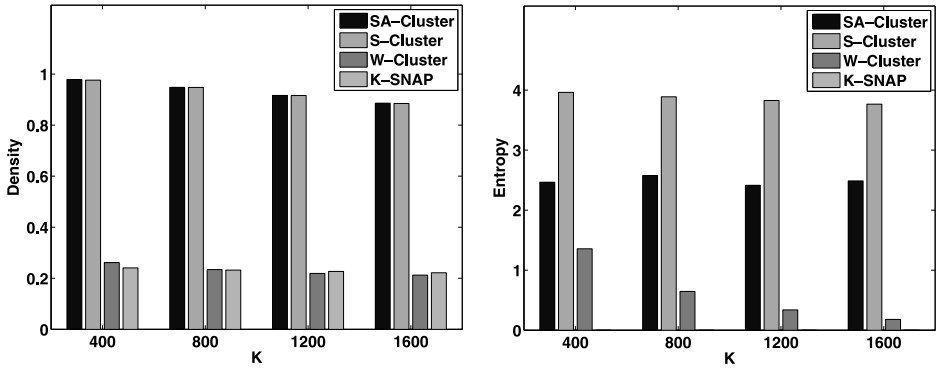
Figures 5(a) and (b) show the density and entropy on the DBLP graph with 5,000 authors with different k values by SA-Cluster, S-Cluster and W-Cluster. As shown in the figures, SA-Cluster achieves a much lower entropy than S-Cluster, with slightly lower density. W-Cluster achieves an even lower entropy 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 conclude that it is hard to use the weighted distance function in W-Cluster to achieve a good balance between attribute similarity and structural similarity. Since k-SNAP strictly enforces the attribute homogeneity in each cluster, with 3 different 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 0.12 (much lower than that of the other three methods) and an entropy of 0.

Figures 6(a) and (b) show the density and entropy on DBLP with 84,170 authors by the four methods when we set $k = 400, 800, 1200, 1600$. These two figures have a similar trend with Figures 5(a)–(b). SA-Cluster achieves similar high density values (> 0.90) with S-Cluster, but with much lower entropy. W-Cluster and k-SNAP achieve very low entropy (the entropy by k-SNAP is 0), but with very low density values at 0.2–0.3. The comparison on both density and entropy demonstrates that our proposed



(a) density (b) entropy

Fig. 5. Cluster quality comparison on DBLP 5,000 authors.



(a) density (b) entropy

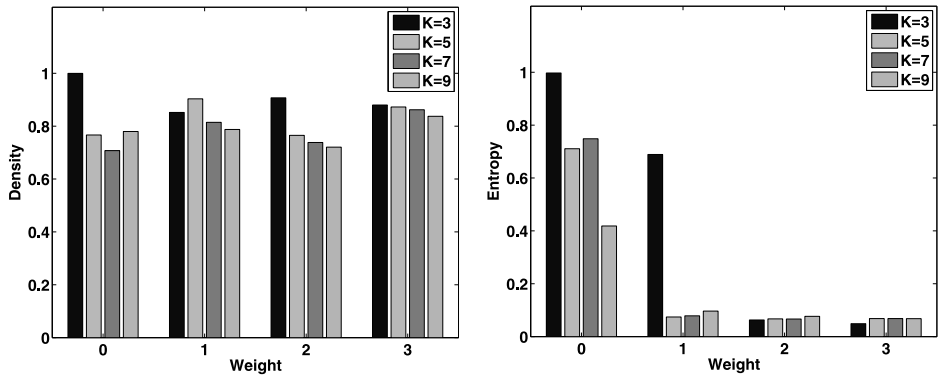
Fig. 6. Cluster quality comparison on DBLP 84,170 authors.

method SA-Cluster achieves a very good balance between the structural cohesiveness and attribute homogeneity.

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 7(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 much. This experiment demonstrates that introducing the attribute similarity into the clustering objective will not downgrade the intra-cluster cohesiveness.

6.4 Clustering Convergence

Tables I–III 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 and cluster refinement. As we can see in Tables I and III, both density and entropy improve as the attribute weights and clusters are computed iteratively. The biggest improvement is usually achieved at iteration 2. Such improvements demonstrate that our weight self-adjustment



(a) density (b) entropy

Fig. 7. Quality vs. different attribute weights on political blogs.

Table I. Cluster Quality in Each Iteration on Political Blogs

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 II. Cluster Quality in Each Iteration on DBLP 5,000 Authors

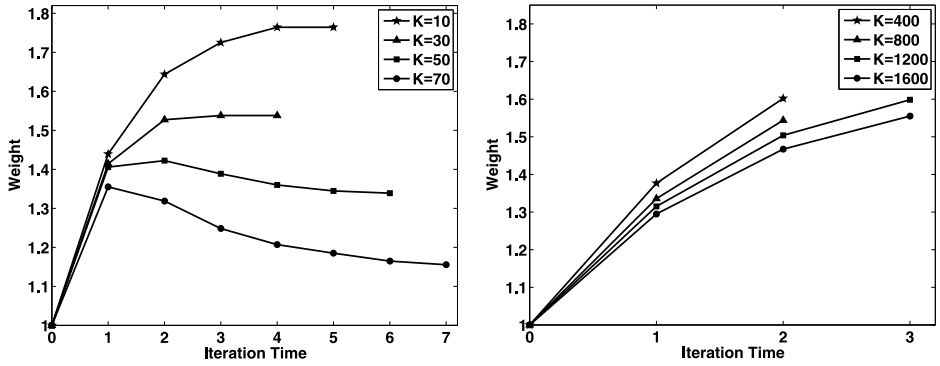
iter#	k = 10		k = 30		k = 50		k = 70	
	D	E	D	E	D	E	D	E
1	0.55	3.12	0.48	2.79	0.47	2.55	0.46	2.43
2	0.55	3.20	0.43	2.49	0.41	2.00	0.38	1.63
3	0.60	3.18	0.46	2.46	0.37	1.99	0.37	1.54
4	0.63	3.23	0.46	2.44	0.38	1.96	0.37	1.53
5	0.63	3.22			0.38	1.96	0.37	1.54
6					0.38	1.96	0.38	1.52
7							0.38	1.52

mechanism is very effective at improving the cluster quality. Similar effects can be observed in Table II on the DBLP graph with 5,000 authors, although entropy gains more improvements while density sacrifices a little bit iteratively.

Figures 8(a) and (b) show 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 Figure 8(a), 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, for example, $k = 10$, 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. In Figure 8(b), the weight updates for different k values have a similar increasing trend.

Table III. Cluster Quality in Each Iteration on DBLP 84,170 Authors

iter#	k = 400		k = 800		k = 1,200		k = 1,600	
	D	E	D	E	D	E	D	E
1	0.72	3.56	0.69	3.34	0.67	3.20	0.65	3.09
2	0.98	3.03	0.95	3.08	0.91	3.08	0.87	3.08
3	0.98	2.47	0.95	2.58	0.92	2.64	0.89	2.69
4					0.92	2.41	0.89	2.49



(a) 5,000 Authors (b) 84,170 Authors

Fig. 8. Weight updates on DBLP dataset.

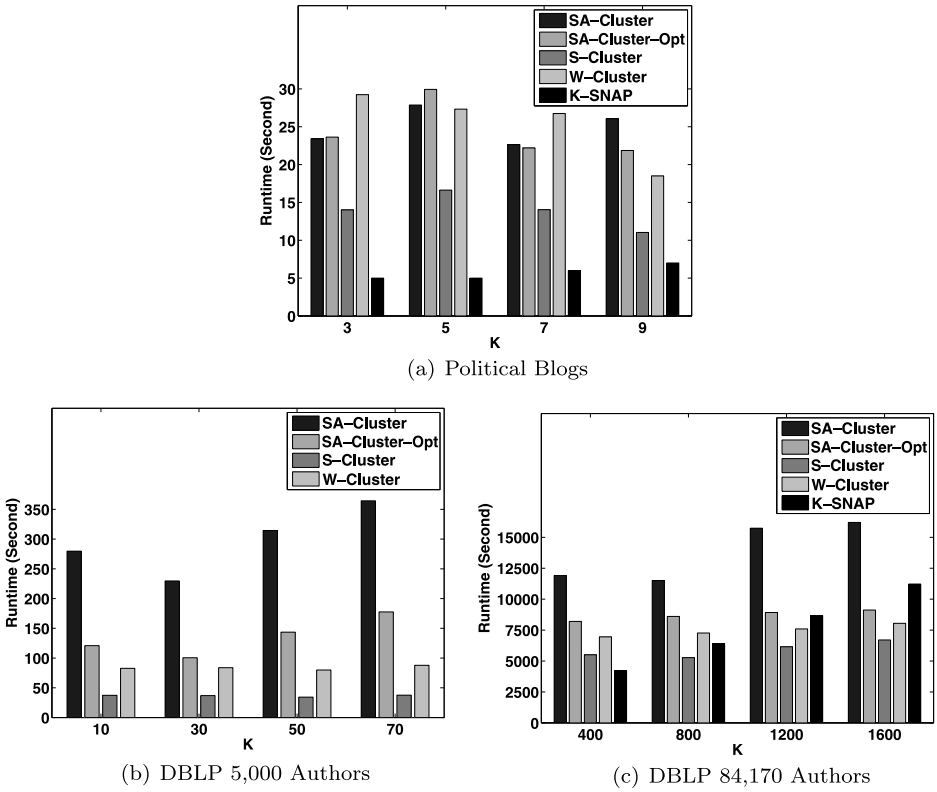
6.5 Clustering Efficiency Evaluation

In this experiment, we compare the efficiency of different clustering algorithms in Figures 9(a)–(c) on Political Blogs, DBLP 5,000 and 84,170 authors respectively. Figure 9(a) shows that all methods have a short clustering runtime on Political Blogs due to its small scale. For both DBLP graphs shown in Figures 9(b) and (c), we can observe that S-Cluster is the most efficient. SA-Cluster is usually 3 to 6 times slower than S-Cluster, as it iteratively computes the random walk distance and performs clustering, while S-Cluster computes random walk only once. In addition, we observe that SA-Cluster-Opt significantly reduces the runtime cost of SA-Cluster to around 1/2–2/3, while achieving the same clustering quality. This result shows that, with the Neumann series technique, SA-Cluster-Opt only causes a relatively small overhead compared with S-Cluster.

6.6 DBLP Case Study

In this experiment, we examine some details of our clustering results on DBLP data with 5, 000 authors when $k = 70$. Table IV shows 4 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 have close collaborations with the selected authors in each cluster due to space limit. Table V shows the keywords and their probabilities which represent the primary topic for each cluster.

Clusters 1−4 contain four groups of authors who work on “database systems,” “frequent pattern mining,” “graphical model learning,” and “information retrieval and relevance feedback,” respectively. Interestingly, each cluster not only contains authors who have close coauthor relationships, it also contains authors who work on the same



(a) Political Blogs

(b) DBLP 5,000 Authors

(c) DBLP 84,170 Authors

Fig. 9. Clustering efficiency.

topic but never collaborate. For example, both *Jiawei Han* and *Mohammed J. 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 the 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, *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.

6.7 Additional Tests on Random Walk Distance Computation

6.7.1 Efficiency Comparison on Random Walk Distance Computation. In this experiment, we compare the efficiency of our proposed random walk computation technique based on Neumann series (Equation (29) in Section 5) with the state-of-the-art technique of fast random walk with restart (RWR) proposed by Tong et al. [2006]. The fast RWR package is downloaded from the author's website and run with default parameters. Table VI shows the runtime cost of both methods in seconds. As we can observe from Table VI, to compute the random walk distance matrix R, our Neumann series based method is much more efficient than fast RWR. We find the advantage of Tong et al. [2006] is in the case if a user is only interested in the random walk score vector of

Table IV. Clusters of Authors from DBLP

Cluster 1 (DB)	**Cluster 2 (DM)**	**Cluster 3 (ML)**	**Cluster 4 (IR)**
François Bancilhon	Gagan Agrawal	Lawrence Carin	James Allan
Alexandros Biliris	Francesco Bonchi	Honghua Dai	Javed A. Aslam
José A. Blakeley	Guozhu Dong	Pedro Domingos	Chris Buckley
Klemens Böhm	Fosca Giannotti	Lise Getoor	Charles L. A. Clarke
Nicolas Bruno	Jiawei Han	Zoubin Ghahramani	W. Bruce Croft
Jan Van den Bussche	Ruoming Jin	Michael I. Jordan	Norbert Fuhr
Michael J. Carey	Jinyan Li	Daphne Koller	A. G. Hauptmann
Sharma Chakravarthy	Wagner Meira Jr.	Pat Langley	Rong Jin
Surajit Chaudhuri	Giuseppe Manco	Yuchang Lu	Joemon M. Jose
Peter Dadam	Richard R. Muntz	Avi Pfeffer	David R. Karger
David J. DeWitt	Siegfried Nijssen	Dan Roth	Mounia Lalmas
C. A. Galindo-Legaria	Srinivasan Parthasarathy	Padhraic Smyth	Victor Lavrenko
Georges Gardarin	Dino Pedreschi	Henry Tirri	Donald Metzler
Goetz Graefe	Jian Pei	Volker Tresp	Alistair Moffat
Theo Härder	Anthony K. H. Tung	Eric P. Xing	Jian-Yun Nie
Won Kim	Adriano Veloso	Kai Yu	Douglas W. Oard
David Maier	Jason Tsong-Li Wang	Shipeng Yu	Jan O. Pedersen
Bernhard Mitschang	Limsoon Wong		Tetsuya Sakai
Vivek R. Narasayya	Stefan Wrobel		Mark Sanderson
Erich J. Neuhold	Ke Wang		Rohini K. Srihari
Z. Meral Özsoyoglu	Xifeng Yan		John Tait
Arnon Rosenthal	Jiong Yang		Xiaojun Wan
Kyu-Young Whang	Philip S. Yu		S. K. Michael Wong
	Osmar R. Zaïane		Chengxiang Zhai
	Mohammed J. Zaki		Justin Zobel

a single vertex, that method can return the random walk score vector of the query vertex very efficiently. For example, on the DBLP graph with 5,000 authors, the time to query one random walk score vector is 0.09 seconds; on the DBLP graph with 84,170 authors, the time to query one random walk score vector is 6.77 seconds. But when we compute the full random walk distance matrix R, the online query procedure in Tong et al. [2006] has to be invoked n times for the n vertices in a graph, which makes the computational cost nontrivial. For example, on the DBLP graph with 84,170 authors, the fast RWR method is 243 times slower than our method. Please note that according to our clustering algorithm in Section 4, it is necessary to compute the full random walk distance matrix R.

6.7.2 Handling the Noninvertible Matrix. In Section 5.2, we discuss the case when the matrix $C = I - (1-c)P_A$ is not invertible and propose a full-rank approximation technique. In this experiment, we test the quality of this approximation technique. In particular, based on the DBLP graph with 5,000 authors, we artificially create the noninvertible $n \times n$ matrix C with rank $n-1, n-2, \ldots, n-5, n-50, n-100$, where n is the number of authors in the graph. We use the original definition of random walk distance in Equation (11), $R_A^l = \sum_{\gamma=0}^{l} c(1-c)^\gamma P_A^\gamma$, to compute the true random walk distance matrix R_{true}. Then based on the full-rank approximation method, we get a full rank matrix C_n which approximates C. According to Equation (29) we compute

Table V. Primary Topic of Each Cluster

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

Table VI. Efficiency Comparison on Random Walk Distance Computation

Dataset	**Neumann Series Based**	**Fast RWR** [Tong et al. 2006]
Political Blogs	10.39	29.68
DBLP 5,000	25.50	436.11
DBLP 84,170	2340.91	569,614.20

Table VII. Quality Comparison on Random Walk Distance Computation

Matrix Rank	**Full-Rank Approximation**	**Pseudoinverse**
$n-1$	6.80E-07	2.53E-01
$n-2$	6.31E-06	3.75E-01
$n-3$	9.35E-06	4.52E-01
$n-4$	1.60E-05	5.14E-01
$n-5$	1.74E-05	5.66E-01
$n-50$	2.64E-05	9.77E-01
$n-100$	6.23E-05	1.02

the approximate random walk distance matrix R_{approx} from C_n. The matrix difference between R_{true} and R_{approx} is measured by the Frobenius norm of $\Delta R = R_{true} - R_{approx}$.

$$\|\Delta R\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} \Delta R_{ij}^2}. \tag{38}$$

To measure the relative error, we compute the Frobenius norm of R_{true} similarly. Then the relative error $\frac{\|\Delta R\|_F}{\|R_{true}\|_F}$ is computed.

Table VII shows that the relative error of R_{approx} vs. R_{true} by our full-rank approximation method is in the scale of 10^{-7}–10^{-5}. The results show that R_{approx} is very close

to R_{true}, which demonstrates the effectiveness of the full-rank approximation method. For comparison purpose, we also tested the pseudoinverse technique [Penrose 1956] to compute the inverse of a noninvertible matrix C and reported the quality of the random walk matrix derived from this method. As we can see from the table, the relative error of the pseudoinverse technique is at least four orders of magnitude higher than that of the full-rank approximation technique.

7. DISCUSSIONS

7.1 Overhead Analysis

In Sections 3 and 4, we introduced how structural and attribute similarities are combined into a unified neighborhood random walk distance by constructing an attribute augmented graph. In this part, we will analyze the computational overhead incurred by the attribute augmented graph.

Assume there are N structure vertices in G and the average degree of a vertex is d. Assume each vertex is associated with m attributes. According to the definition of attribute augmented graph, we know that $N \times m$ attribute edges need to be added to the original graph to form the attribute augmented graph. Therefore, there are totally $N \times d$ structure edges and $N \times m$ additional attribute edges in the attribute augmented graph. When m is larger than d, the number of attribute edges will be larger than the number of structure edges. To compute the random walk distance involving attribute edges, the computational overhead could be nontrivial when $m \gg d$.

In our proposed approach, we also assume that the attributes are categorical ones, that is, an attribute a_i has n_i possible values. Thus n_i attribute vertices can be added to represent the n_i different values of the attribute a_i. To make our approach applicable to a wide range of applications, we will discuss how to handle a large number of attributes (i.e., when m is large) as well as continuous attributes with preprocessing techniques in the following.

7.2 Handling a Large Number of Attributes and Continuous Attributes

As we have discussed, when the number of attributes is large, the computational overhead for computing the random walk distance on the attribute augmented graph is nontrivial. In this case, we can perform correlation analysis to detect correlation between attributes and then perform dimensionality reduction to retain a smaller set of orthogonal dimensions. Widely used dimensionality reduction techniques such as principal component analysis (PCA) and multifactor dimensionality reduction (MDR) can be used to create a mapping from the original space to a new space with fewer dimensions. According to the mapping, we can compute the new attribute values of a vertex based on the values of its original attributes. Then we can construct a more compact attribute augmented graph in the new feature space and perform graph clustering.

To handle continuous attributes, discretization can be applied to convert them to nominal features. Typically the continuous values are discretized into K partitions of an equal interval (equal width) or K partitions each with the same number of data points (equal frequency). In our experiment, there is an attribute "prolific" for each author in the DBLP bibliographic graph indicating whether the author is prolific. If we use the number of publications to measure the prolific value of an author, then "prolific" is a continuous attribute. According to the distribution of the publication number in DBLP, we discretized the publication number into 3 partitions: authors with < 5 papers are labeled as low prolific, authors with ≥ 5 but < 20 papers are prolific, and the authors with ≥ 20 papers are tagged as highly prolific.

8. RELATED WORK

In literature, many graph clustering techniques have been proposed which mainly focused on the topological structures based on various criteria including normalized cuts [Shi and Malik 2000], modularity [Newman and Girvan 2004] or structural density [Xu et al. 2007]. 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. [2008] 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 intracluster 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. [2007] 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. [2009] 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. [2008] proposed a graph summarization method using the MDL principle. Tsai and Chui [2008] 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. Cai et al. [2005] proposed an algorithm 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 [2002] designed a measure called SimRank, which defines the similarity between two vertices in a graph by their neighborhood similarity. Pons and Latapy [2006] proposed to use short random walks of length l to measure the similarity between two vertices in a graph for community detection. Tong et al. [2006] designed an algorithm for fast random walk computation. Other studies which use random walk with restarts include connection subgraph discovery [Faloutsos et al. 2004] and center-piece subgraph discovery [Tong and Faloutsos 2006]. Liu et al. [2008] proposed to use random walk with restart to discover subgraphs that exhibit significant changes in evolving networks.

9. CONCLUSIONS

In this article, we study the problem of large graph clustering by achieving a balance between 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 intracluster 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. Optimization techniques on matrix computation based on Matrix Neumann Series are designed to improve efficiency in calculating the random walk distance. Experimental

results on large real graphs demonstrate that our method achieves a very good balance between structural and attribute similarities.

APPENDIX

PROOF OF THEOREM 1. First, 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

$$P_V^{k+1}(v_j, v_p) = P_V^{k+1}(v_j, v_q),\ \forall v_j \in V \tag{39}$$

Second, 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 Equation (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

$$\sum_{v_c = v_i, v_p, v_q} C^k(v_i, v_c) \cdot C(v_c, v_p) - \sum_{v_c = v_i, v_p, v_q} C^k(v_i, v_c) \cdot C(v_c, v_q) > 0. \tag{40}$$

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

$$C^k(v_i, v_i) \cdot (C(v_i, v_p) - C(v_i, v_q)) > 0 \tag{41}$$

$$(C^k(v_i, v_p) - C^k(v_i, v_q)) \cdot (C(v_p, v_p) - C(v_q, v_p)) \geq 0. \tag{42}$$

Combining Equations (41) and (42), we can prove Equation (40). 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

$$C^l(v_c, v_p) \geq C^l(v_c, v_q), \forall v_c \neq v_p, v_q. \tag{43}$$

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

$$(P_V^{m_i} C^{n_i})(v_i, v_p) > (P_V^{m_i} C^{n_i})(v_i, v_q). \tag{44}$$

We have

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

and

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

Since we have proven Equations (39) and (43), we can prove Equation (44), 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:

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

ACKNOWLEDGMENTS

We thank Yizhou Sun for providing us the DBLP data with topic information. We also thank Zheng Liu for his valuable comments.

REFERENCES

AGRAWAL, R., GEHRKE, J., GUNOPULOS, D., AND RAGHAVAN, P. 1998. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data (SIGMOD'98)*. 94–105.

APOSTOL, T. M. 1967. *Calculus, Vol. 1: One-Variable Calculus, with an Introduction to Linear Algebra* 2nd Ed. Wiley.

BOTTON, L. AND BENGIO, Y. 1994. Convergence properties of the k-means algorithms. In *Proceedings of Advances in Neural Information Processing Systems 7 (NIPS'94)*. 585–592.

CAI, D., SHAO, Z., HE, X., YAN, X., AND HAN, J. 2005. Mining hidden community in heterogeneous social networks. In *Proceedings of the Workshop on Link Discovery: Issues, Approaches and Applications (LinkKDD'05)*. 58–65.

DESCARTES, R. 1954. *The Geometry of René Descartes*. Dover Publications.

ESTER, M., KRIEGEL, H.-P., SANDER, J., AND XU, X. 1996. A density-based algorithm for discovering clusters in large spatial databases. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'96)*. 226–231.

FALOUTSOS, C., MCCURLEY, K., AND TOMKINS, A. 2004. Fast discovery of connection subgraphs. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*. 118–127.

FINE, B. AND ROSENBER, G. 1997. *The Fundamental Theorem of Algebra (Undergraduate Texts in Mathematics)*. Springer-Verlag.

GIBSON, D., KLEINBERG, J., AND RAGHAVAN., P. 1998. Inferring web communities from link topology. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*. 225–234.

HINNEBURG, A. AND KEIM, D. A. 1998. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'98)*. 58–65.

HOFMANN, T. 1998. Probabilistic latent semantic indexing. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*. 50–57.

JEH, G. AND WIDOM, J. 2002. SimRank: A measure of structural-context similarity. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*. 538–543.

KAUFMAN, L. AND ROUSSEEUW, P. J. 1987. Clustering by means of medoids. In *Statistical Data Analysis Based on the L1 Norm*. 405–416.

LIU, Z., YU, J. X., KE, Y., LIN, X., AND CHEN, L. 2008. Spotting significant changing subgraphs in evolving graphs. In *Proceedings of the 2008 International Conference on Data Mining (ICDM'08)*.

MANNING, C. D., RAGHAVAN, P., AND SCHÜTZE, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

NAVLAKHA, S., RASTOGI, R., AND SHRIVASTAVA, N. 2008. Graph summarization with bounded error. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data (SIGMOD'08)*. 419–432.

NEWMAN, M. E. J. AND GIRVAN, M. 2004. Finding and evaluating community structure in networks. *Phys. Rev. E 69, 026113*.

NG, R. AND HAN, J. 1994. Efficient and effective clustering method for spatial data mining. In *Proceedings of the International Conference on Very Large Data Bases (VLDB'94)*. 144–155.

PENROSE, R. 1956. On best approximate solution of linear matrix equations. *Math. Proc. Cambridge Phil. Soc. 52*, 17–19.

PONS, P. AND LATAPY, M. 2006. Computing communities in large networks using random walks. *J. Graph Algor. Appli. 10,* 2, 191–218.

SHI, J. AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE Trans. Patt. Anal. Mach. Intell*. *22*, 8, 888–905.

STRANG, G. 2005. *Linear Algebra and its Applications*. Brooks Cole.

SUN, J., FALOUTSOS, C., PAPADIMITRIOU, S., AND YU, P. S. 2007. Graphscope: Parameter-free mining of large time-evolving graphs. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*. 687–696.

SUN, Y., HAN, J., ZHAO, P., YIN, Z., CHENG, H., AND WU, T. 2009. Rankclus: Integrating clustering with ranking for heterogenous information network analysis. In *Proceedings of the International Conference on Extending Database Technology (EDBT'09)*. 565–576.

TIAN, Y., HANKINS, R. A., AND PATEL, J. M. 2008. Efficient aggregation for graph summarization. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data (SIGMOD'08)*. 567–580.

TONG, H. AND FALOUTSOS, C. 2006. Center-piece subgraphs: problem definition and fast solutions. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06)*. 404–413.

TONG, H., FALOUTSOS, C., AND PAN, J.-Y. 2006. Fast random walk with restart and its applications. In *Proceedings of the International Conference on Data Mining (ICDM'06)*. 613–622.

TSAI, C.-Y. AND CHUI, C.-C. 2008. Developing a feature weight self-adjustment mechanism for a k-means clustering algorithm. *Computat. Statis. Data Anal. 52*, 4658–4672.

XU, X., YURUK, N., FENG, Z., AND SCHWEIGER, T. A. J. 2007. Scan: A structural clustering algorithm for networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*. 824–833.

ZHAI, C., VELIVELLI, A., AND YU, B. 2004. A cross-collection mixture model for comparative text mining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*. 743–748.

ZHOU, Y., CHENG, H., AND YU, J. X. 2009. Graph clustering based on structural/attribute similarities. In *Proceedings of the International Conference on Very Large Data Bases (VLDB'09)*. 718–729.

Received January 2010; revised May 2010; accepted July 2010