Information Sciences 314 (2015) 77–99

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Dense community detection in multi-valued attributed networks

Xin Huang [a,b,*], Hong Cheng [a], Jeffrey Xu Yu [a]

[a] The Chinese University of Hong Kong, Hong Kong, China
[b] University of British Columbia, Canada

ARTICLE INFO

Article history:
Received 25 September 2014
Received in revised form 23 February 2015
Accepted 30 March 2015
Available online 4 April 2015

Keywords:
Community detection
Dense subgraph
Attributed graph
Random walk with restart

ABSTRACT

The proliferation of rich information available for real world entities and their relationships gives rise to a general type of graph, namely *multi-valued attributed graph*, where graph vertices are associated with a number of attributes and a vertex may have multiple values on an attribute. It would be useful if we can cluster such graphs into densely connected components with homogeneous attribute values. Recent work has studied graph clustering in attributed graphs considering the full attribute space. However, full space clustering often leads to poor results due to irrelevant attributes.

In this paper, we study subspace clustering in multi-valued attributed graph and propose an algorithm SCMAG for community detection. Our algorithm uses a cell-based subspace clustering approach and identifies cells with dense connectivity in the subspaces. Random walk with restart is used to measure the structural connectivity and attribute similarity. An indexing scheme is designed to support efficiently calculating cell connectivity from random walk scores. We also propose a new cell combining strategy on dimensions of categorical attributes and a novel mechanism to handle multi-valued attributes. Experimental results on IMDB data and bibliographic data demonstrate that SCMAG significantly outperforms the state-of-the-art subspace clustering algorithm and attributed graph clustering algorithm. Case studies show SCMAG can find dense communities with homogeneous properties under different subspaces.

1. Introduction

Graph clustering has attracted a lot of attention recently in the literature. The goal of graph clustering is to group densely connected vertices into a cluster. Graph clustering has found broad applications in community detection, image segmentation, identification of functional modules in large protein–protein interaction networks, etc. Nowadays with rich information available for real world entities and their relationships, graphs can be built in which vertices are associated with a set of attributes describing the properties of the vertices. Such attributes can be numerical or categorical. For some attribute, an entity can have more than one value on that attribute. For example, the *genre* of a movie can be both "*drama*" and "*crime*"; the *research topics* of a researcher can be "*database*", "*data mining*" and "*bioinformatics*". Multi-valued attributes are very common in many real world data. This gives rise to a new and more general type of graph, called *multi-valued attributed graph*.

* Corresponding author at: University of British Columbia, Canada.
E-mail addresses: xhuang@se.cuhk.edu.hk (X. Huang), hcheng@se.cuhk.edu.hk (H. Cheng), yu@se.cuhk.edu.hk (J.X. Yu).

http://dx.doi.org/10.1016/j.ins.2015.03.075
0020-0255/© 2015 Elsevier Inc. All rights reserved.

Traditional graph clustering methods [29,21,34,26,27] mainly focus on the connections in a graph and try to achieve a dense subgraph within a cluster. But these methods do not consider attribute information associated with graph vertices. On the other hand, a graph summarization method [31] partitions a graph according to attribute values, but does not enforce dense connections within a partition.

There have been some recent studies [6,37] on clustering attributed graph, i.e., SA-Cluster and its fast version Inc-Cluster, which partition the graph into several densely connected components with similar attribute values. SA-Cluster finds non-overlapping clusters in the full attribute space. Although SA-Cluster differentiates the importance of attributes with an attribute weighting strategy, it cannot get rid of irrelevant attributes completely. As graph vertices may have many attributes, the high-dimensional clusters are hard to interpret, or there is even no significant cluster with homogeneous attribute values in the full attribute space. If an attributed graph is projected to different attribute subspaces, various interesting clusters embedded in subspaces can be discovered which, however, may not exhibit in the full attribute space. In this paper, we study subspace clustering on multi-valued attributed graph, and discover clusters embedded in subspaces. Such subspace clusters should not only have homogeneous attribute values but also have dense connections, i.e., correspond to communities with homogeneous properties. The detected clusters in multi-valued attributed graphs can overlap, as the nodes may belong to multiple clusters in different subspaces. In contrast, non-overlapping community detection partitions the network into several disjoint clusters. Thus, overlapping communities are more common and natural than non-overlapping communities in attributed graphs. For example, an individual in a social network belongs to many social circles corresponding to different relationships, such as friends, schoolmates, family, research community and so on. Let us look at an example to illustrate the motivation for this subspace clustering problem.

Fig. 1(a) shows a coauthor network with three attributes {*Topic, Affiliation, Country*}. A vertex represents an author and an edge represents the coauthor relationship between two authors. The attributes and their possible values are listed in Table 1. *Topic* is a multi-valued attribute and an author can have one or more topics, e.g., author v_4. The problem is how to partition the coauthor network into clusters with close collaborations and homogeneous attribute values.

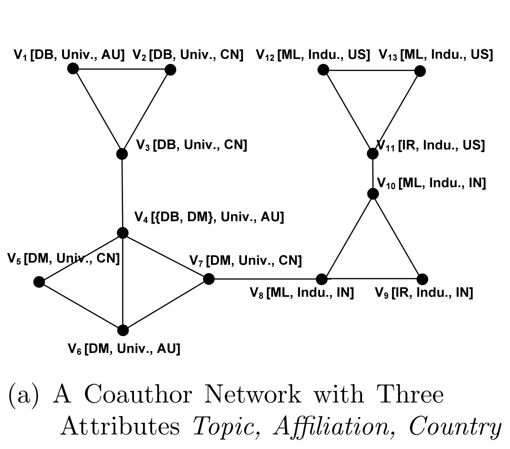
If we apply SA-Cluster [6], the coauthor network is partitioned into 4 clusters, as shown in Fig. 1(b). But this clustering result is not satisfactory, because (1) the attribute values in the same cluster are still quite different, if considering the full attribute space; and (2) v_9 is disconnected from his coauthors and forms a single-node cluster. This is not reasonable. In a high-dimensional attributed graph this problem may become even worse, as it is hard to find clusters with dense connectivity and homogeneous attribute values in the full space.

If we consider subspace clustering, then we can find two clusters under the subspace {*Affiliation, Country*} in Fig. 2(a), and another two clusters under the subspace {*Topic, Affiliation*} in Fig. 2(b). These subspace clusters make much more sense because they not only have homogeneous attribute values in the respective subspace, but also have a cohesive structure within a cluster. Note that, if traditional subspace clustering methods, e.g., CLIQUE [2] and ENCLUS [5] are applied, under the subspace {*Affiliation, Country*} we will have two more clusters $\{v_1, v_4, v_6\}, \{v_2, v_3, v_5, v_7\}$ respectively with attribute values [*Univ., AU*] and [*Univ., CN*]. However, we can easily find these two clusters have very sparse connections, thus not closely collaborating groups.

This example shows neither the recent attributed graph clustering algorithms [6,37] which consider the full attribute space, nor the traditional subspace clustering algorithms [2,5] which completely ignore the structural connectivity are suitable to solve the problem of multi-valued attributed graph clustering. The unique challenges in this problem include the following:

1. how to discover subspaces under which densely connected clusters with homogeneous attribute values are embedded? For example, there are meaningful clusters under the two subspaces shown in Fig. 2(a) and (b), but there is no cluster with dense structural connectivity under the subspace {*Topic, Country*};
2. how to properly handle categorical attributes and multi-valued attributes. Traditional subspace clustering algorithms, e.g., CLIQUE [2] and ENCLUS [5], only handle numerical attributes. But in real data categorical and multi-valued attributes are very common, for example, v_4 has two topics as *DB* and *DM*. Which cluster should v_4 belong to in Fig. 2(b)?
3. how to enforce both structural connectivity and attribute similarity requirements, like in the clusters in Fig. 2 (a) and (b)?

There have been some recent studies which combine attribute subspace clustering and dense subgraph mining on a graph with feature vectors, e.g., CoPaM [19], GAMer [12,13], DB-CSC [11]. CoPaM and GAMer follow a cell-based subspace clustering approach to find clusters that show high similarity in a feature subspace and are densely connected in the given graph, while DB-CSC takes a density-based approach to find subspace clusters. They have the different limitations: (1) CoPaM and GAMer do not have a cell merging strategy for adjacent cells, and the formation of a cell depends on the initial vertex to start with; (2) CoPaM strictly requires all nodes in a cluster have the same value on each attribute in the subspace. GAMer and DB-CSC use a single parameter *maximal width w* to control the value difference on *all the attributes* in the concerned subspace within a cell. But for different attributes, which can be categorical or numerical ones, it is hard to set a uniform threshold to control the attribute value differences; (3) The time complexity of GAMer and DB-CSC increases exponentially with the number of vertices in the input graph, which can hardly scale with large graphs. Even though adjusting by various parameter settings, GAMer is proven to be too slow to generate results in our experiments; and (4) According to the experimental results reported in [12,13,11], only clusters with small size are found by GAMer and DB-CSC on real datasets, i.e., the average size of the found clusters is only around 10 in a graph with 10k vertices. Similarly, according to [19], the largest found subgraph



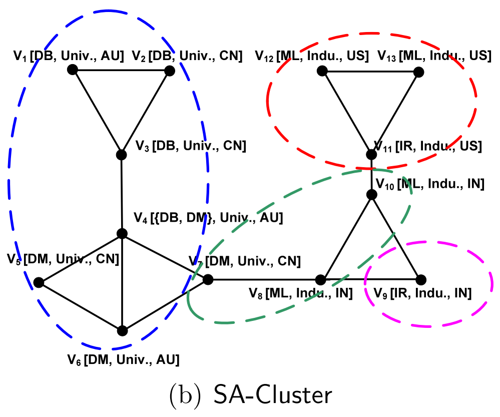
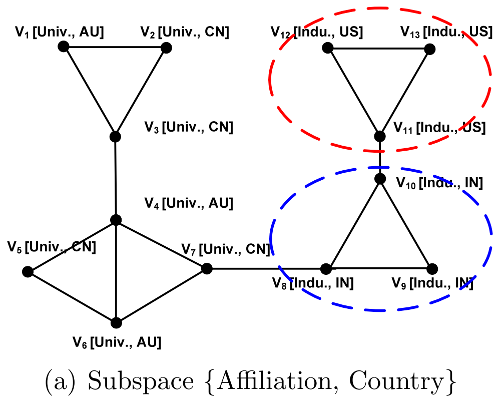


Fig. 1. Running example.

Table 1
Attributes and domains.

Attribute	Domain
Topic	DB, DM, ML, IR
Affiliation	University, Industry
Country	Australia, China, India, United States



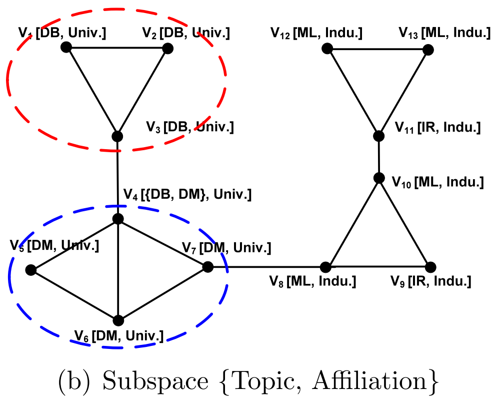


Fig. 2. Subspace clustering on the coauthor network.

by CoPaM has 18 vertices in a graph with 1942 vertices. Overall, the efficiency challenge of detecting large communities by CoPaM, GAMer and DB-CSC have not been verified yet. Large communities naturally exist in many real-world networks, e.g. in terms of research areas, the research community of data mining in the coauthor network would contain far more than 1000 individuals.

Different from these methods, in this paper we propose a novel subspace clustering algorithm on attributed graphs. Our clustering framework is similar to that in ENCLUS [5]. We first find interesting subspaces with good clustering using an entropy-based method. Then we identify cells with dense structural connectivity in the grid under the subspaces. Clusters are formed by merging adjacent cells. Random walk with restart on an attribute augmented graph is used to measure the structural connectivity between vertices as well as the attribute similarity. Our main contributions include:

- We study the problem of graph subspace clustering on attributed graphs. We give a cell-based subspace clustering definition, and propose a novel algorithm, called SCMAG (**S**ubspace **C**lustering on **M**ulti-valued **A**ttributed **G**raph). With our formulation, we can discover more meaningful clusters which are closely connected and exhibit high attribute similarity under some subspaces. Subspace clusters may overlap and they provide multiple angles to exhibit cluster patterns under different subspaces.

- We use random walk with restart to unify the structural closeness and attribute similarity into a single measure. Based on the random walk score, we design a novel cell combining strategy on dimensions of categorical attributes and an effective mechanism to handle multi-valued attributes.
- We consider both structural connectivity and attribute similarity in the subspace clustering process. We design an index scheme to speedup calculating the connectivity of a cluster. An upper bound is also derived for efficiently pruning clusters with low connectivity.
- We conducted extensive experiments on real datasets to evaluate the performance of our method in terms of structural density and attribute value similarity, and demonstrate its superiority over existing methods. We also show some interesting clusters we discovered in the case study.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 introduces the preliminary concepts and describes our random walk based closeness measure. Our subspace clustering algorithm SCMAG is described in Section 4. We present extensive experimental results in Section 5. Finally Section 6 concludes our paper.

2. Related work

Subspace clustering [2,22,18,23] has been studied extensively since late 1990s. CLIQUE [2] was one of the first subspace clustering algorithms which uses an APRIORI search framework to find dense subspace clusters. ENCLUS [5] is an extension of CLIQUE which proposes to use entropy to find subspaces with good clustering. DOC [25] is a density-based projective clustering algorithm which uses a Monte Carlo procedure to approximate with high probability an optimal projective cluster. PROCLUS [1] focuses on the clustering results to find appropriate subspace such that the inner-cluster points are close. These subspace clustering algorithms only consider numerical attributes but completely ignore the relationships between objects. Thus they are not suitable to cluster attributed graphs. A comprehensive survey of subspace clustering algorithms proposed in recent years can be found in [30].

Traditional graph clustering methods [28] aim to have a dense subgraph within a cluster. Many graph clustering algorithms have been proposed based on various criteria including normalized cuts [29], modularity [21], structural density [34], weighted kernel k-means [10] or stochastic flows [26]. Satuluri et al. [27] study how to sparsify a graph for scalable clustering without sacrificing quality. Recently, several new community model based different types of dense subgraph are proposed for community search, such as quasi-clique community model [8] and k-truss community model [14]. But these methods do not consider attribute information associated with graph vertices. Some recent studies consider summarizing or clustering attributed graphs in the full attribute space. Tian et al. [31] proposed OLAP-style aggregation approaches to summarize large graphs by grouping nodes based on user-selected attributes. This method achieves homogeneous attribute values within clusters, but ignores the structure connectivity. Recently, Zhou et al. have proposed a graph clustering algorithm, SA-Cluster [6] and its fast version Inc-Cluster [37], based on both structural and attribute similarities. [4] proposed a spectral method for clustering on attributed graphs. A parameter-free method [15] is proposed to identify dense clusters and outliers in attributed graphs. Long et al. [17] proposed a collective factorization model for multi-type relational data clustering. Yang et al. [35] studied the problem of detecting overlapping communities in networks with node attributes. However, these algorithms consider clustering in the full attribute space, thus it may be hard to find good clusters in high-dimensional attributed graphs.

Random walk with restart (RWR) [33] provides a good relevance measure between two nodes in weighted graphs, and it has wide applications in many domains, such as automatic captioning of images, generalizations to the connection subgraphs, personalized PageRank. Jeh and Widom [16] defined a measure called SimRank, which measures the similarity between two vertices in a graph by their neighborhood similarity. Pons and Latapy [24] proposed a method to use short random walks of length l to calculate the similarity between two vertices for community detection in a graph. Desikan et al. [9] proposed an incremental algorithm to effciently compute PageRank for the dynamic graph by partitioning the graph into a changed part and an unchanged part. [36] studied a new problem of reverse top-k proximity search in graphs based on random walk with restart.

There have been some recent studies which mine cohesive subgraph patterns from a graph with feature vectors, and combine the concepts of dense subgraph mining and subspace clustering. These methods include CoPaM [19], GAMer [12,13], and DB-CSC [11]. Following a cell-based subspace clustering approach, CoPaM and GAMer mine clusters containing all objects falling within a cell, i.e., the attribute values differ by at most a given threshold, with some density constraint. However, neither methods have a cell merging strategy to merge adjacent cells, which is different from our work. The formation of a cell also depends on the initial vertex to start with and its feature vector. DB-CSC takes a density-based approach for finding subspace clusters. Similar to GAMer, DB-CSC uses a single parameter *maximal width w* to control the value difference on *all the attributes* in the concerned subspace within a cell. But for different attributes, which can be categorical or numerical ones, it is hard to set a uniform threshold to control the attribute value differences. Another issue is that the time complexity of GAMer and DB-CSC increases exponentially with the number of vertices in the graph, which can hardly scale with large graphs. In addition, according to the experimental results in [12,13,11], only clusters with small size are found and shown by GAMer and DB-CSC on real datasets, i.e., the average size of the found clusters is only around 10 in a graph with 10k nodes.

3. Preliminary concepts

In this section, we firstly give our problem definition and a brief introduction of ENCLUS, a subspace clustering method for numerical data, on which our graph clustering method is built. Then we describe how to measure the structural connectivity and attribute similarity of graph vertices. We also propose a novel strategy to handle the categorical and multi-valued attributes in graph subspace clustering.

3.1. Problem definition

Definition 3.1 (*Multi-valued Attributed Graph*). A multi-valued attributed graph is denoted as $G = (V, E, \mathcal{A}, w)$, where V is the vertex set, E is the edge set, $\mathcal{A} = \{A_1, \ldots, A_d\}$ is a set of categorial or totally ordered numerical attributes. Each vertex $v \in V$ is associated with an attribute vector $A(v) = [A_1(v), \ldots, A_d(v)]$, where $A_i(v)$ is the attribute value that vertex v has on attribute A_i. The domain of a categorical attribute $A_i \in \mathcal{A}$ contains a finite set of values, denoted as $Dom(A_i) = \{a_{i1}, \ldots, a_{in_i}\}, |Dom(A_i)| = n_i$. If A_i is a multi-valued categorical attribute, $A_i(v)$ contains a set of attribute values from $Dom(A_i)$, i.e., $A_i(v) \subseteq Dom(A_i)$. The domain of a numerical attribute $A_j \in \mathcal{A}$ is $\mathbb{R}$. $w : E \mapsto \mathbb{R}^+$ is an edge weight function. For each $(u, v) \in E$, the edge weight is denoted as $w(u, v)$. We denote the size of vertex set as $|V| = N$.

Many real-world networks can be represented as multi-valued attributed graphs with weighted edges, such as co-author network and IMDB network. Continue with our example in Fig. 1(a), the weight of edge (u, v) in co-author networks can be assigned with the number of co-authored papers pubished by authors u and v, which represents the collaboration strength. The larger weight of $w(u, v)$ is, the stronger relationship between u and v is.

Remark 3.1. In the paper, our proposed subspace clustering method can also handle graphs with multi-valued edge labels, in which the edges can be associated with multiples values in attributes. To handle such a graph, denoted as G, with multi-valued edge labels, we need to construct a new attributed graph G' as follows. For each edge (u, v) in G, a new vertex e_{uv} is inserted into G'. We assign e_{uv} with the same attribute vector of edge (u, v). For two edges (u, v) and (u, w) adjacent to u in G, a new edge (e_{uv}, e_{uw}) is added between two vertices e_{uv}, e_{uw} in G'. Then, G' is a multi-valued attributed graph by Definition 3.1, our subspace clustering methods can be applied on G'.

If an attribute set $\mathcal{S} \subseteq \mathcal{A}$, we say $\mathcal{S}$ is a subspace of $\mathcal{A}$. The problem of *subspace clustering on a multi-valued attributed graph* is to find a set of subspaces $\{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$ with good clustering, where $\mathcal{S}_i \subseteq \mathcal{A}, 1 \leqslant i \leqslant m$. Under each subspace $\mathcal{S}_i$, discover embedded clusters which have both high structural connectivity and high attribute similarity in the concerned subspace. We will define the formal requirements of “good subspace” and “embedded clusters” in Section 4.1.

3.2. ENCLUS: a brief introduction

Our proposed graph subspace clustering algorithm is built based on ENCLUS [5], an entropy-based subspace clustering method on data objects with numerical features. We briefly describe ENCLUS here. It first divides each numerical attribute into intervals like $[l, u)$, where l and u denote the interval lower bound and upper bound, respectively. Then the space is partitioned to form a grid. A cell has the form $\{c_1, c_2, \ldots, c_d\}$ where $c_i = [l_i, u_i)$ is an interval of attribute A_i. The subspace entropy is defined as follows.

Definition 3.2 (*Subspace Entropy*). Given a set of attributes $\mathcal{S} = \{A_1, \ldots, A_k\} \subseteq \mathcal{A}$, the subspace entropy of $\mathcal{S}$ is defined as

$$H(A_1, \ldots, A_k) = -\sum_{a_1 \in Dom(A_1)} \cdots \sum_{a_k \in Dom(A_k)} p(a_1, \ldots, a_k) \log p(a_1, \ldots, a_k) \quad (1)$$

where $p(a_1, \ldots, a_k)$ is the percentage of graph vertices whose attribute value vector is $[a_1, \ldots, a_k]$.

The framework of ENCLUS consists of the following three steps.

1. Find interesting subspaces with good clustering by an entropy-based method;
2. Find clusters in the identified subspace by merging adjacent dense cells;
3. Output the clusters.

The term "good clustering" means that a subspace contains a good set of clusters. Formally the goodness of a subspace is measured by entropy. Given an entropy threshold, a subspace whose entropy is below the threshold is considered to have good clustering. ENCLUS uses a bottom-up approach similar to APRIORI to find good subspaces. It starts with finding one-dimensional subspaces with good clustering. Then it generates the candidate two-dimensional subspaces from the one-dimensional ones and checks the candidates against the raw data to identify those that actually have good clustering. This process is repeated with increasing dimensionalities until no more subspaces with good clustering are found. The correctness of this bottom-up approach is based on the *downward closure property* of entropy, as proved in [5].

Under each good subspace, adjacent dense cells are combined to form a cluster. Two k-dimensional cells are called "*adjacent*" if they have the same attribute value intervals on $k-1$ dimensions, and on the remaining dimension, the interval upper bound of a cell is equal to the interval lower bound of the other cell. Finally the clusters are returned to users.

3.3. A unified closeness measure

ENCLUS only considers the attribute values of data objects, but ignores the connections between them. In our graph subspace clustering problem, to calculate the structural connectivity of vertices, we propose to use random walk with restart to measure the closeness between two vertices. According to the properties of small-world networks, the average shortest distance between two nodes would be significantly small. For example, in the Facebook social network, the average shortest distance was 4.74 in 2011 [3]. This follows from the fact that two "distant" can have a short path flow through hubs. In this case, the number of paths between two nodes would be more effective for measuring their connectivity than the accurate shortest distance. As a result, it is better to use random walk with restart to measure the connectivity of vertices, than simple distance measures, e.g., short distance and nearest neighbors.

Definition 3.3 (*Neighborhood Random Walk*). Let P be the $N \times N$ transition probability matrix of a graph. Given L as the maximum length a random walk can go, $c \in (0,1)$ as the restart probability, the neighborhood random walk matrix is

$$R = \sum_{l=1}^{L} c(1-c)^l P^l \qquad (2)$$

The random walk score of two vertices $v_i, v_j \in V$ is $R(v_i, v_j)$, measuring the probability that v_i can reach v_j within L steps.

For the attributes on graph vertices, we do the following preprocessing. If A_i is a numerical attribute, it is discretized into n_i intervals first. There are some studies, e.g., MAFIA [20], which consider adaptive interval partitioning of numerical attributes in subspace clustering. But this is not the focus of our work. In our more general problem setting, we also consider categorical and multi-valued attributes. We have to address two problems: (1) how to measure the similarity between two categorical values and determine which categorical values are "adjacent" for merging cells with categorical attributes? and (2) if a vertex has multiple values on one attribute, which cell should the vertex be assigned to in the formed grid? To solve these problems, we consider the neighborhood of a vertex to exploit the connections with its neighbors as well as the attribute similarities with its neighbors. We construct the attribute augmented graph similarly as in [6] [32]. The definition is as follows.

Definition 3.4 (*Attribute Augmented Graph*). Given a multi-valued attributed graph denoted as $G = (V, E, \mathcal{A}, w)$ where $\mathcal{A} = \{A_1, \ldots, A_d\}$. The domain of an attribute A_i is denoted as $Dom(A_i) = \{a_{i1}, \ldots, a_{in_i}\}, |Dom(A_i)| = n_i$. An attribute augmented graph is denoted as $G_a = (V \cup V_a, E \cup E_a)$, where $V_a = \{v_{ij} | 1 \leqslant i \leqslant d, 1 \leqslant j \leqslant n_i\}$ is the set of attribute vertices. An attribute vertex $v_{jk} \in V_a$ represents the value a_{jk} on attribute A_j. An edge $(v_i, v_{jk}) \in E_a$ iff $a_{jk} \in A_j(v_i)$, i.e., vertex v_i has value 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.

For instance, we construct an attribute augmented graph shown in Fig. 3(b) for the co-author network in Fig. 3(a). In this attribute augmented graph, we add a set of dummy vertices $V_a\{v_8, v_9, v_10, v_{11}, v_{12}\}$, e.g. v_8, which represents the attribute value "DB" in "Topic". Authors with corresponding attribute values are connected to the two vertices respectively in dashed lines, e.g. the edge between v_1 and v_8, indicating that author v_1 has the value "DB" in the attribute of "Topic". With the newly inserted attribute edges, authors who are originally connected become more densely conncetted if they share common attribute values, e.g., v_5 and v_7.

With such a graph augmentation, we express the attribute similarity as vertex closeness in the graph: two vertices which share an attribute value are connected by a common attribute vertex. In the attribute 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. We simulate a length-L neighborhood random walk on the attribute augmented graph, which goes through both structure edges and attribute edges. The random walk score effectively combines structural closeness and attribute similarity into a unified closeness measure between graph vertices. The transition matrix P_A of the attribute augmented graph is a $|V \cup V_a| \times |V \cup V_a|$ matrix. We denote $|V_a| = N_A$.

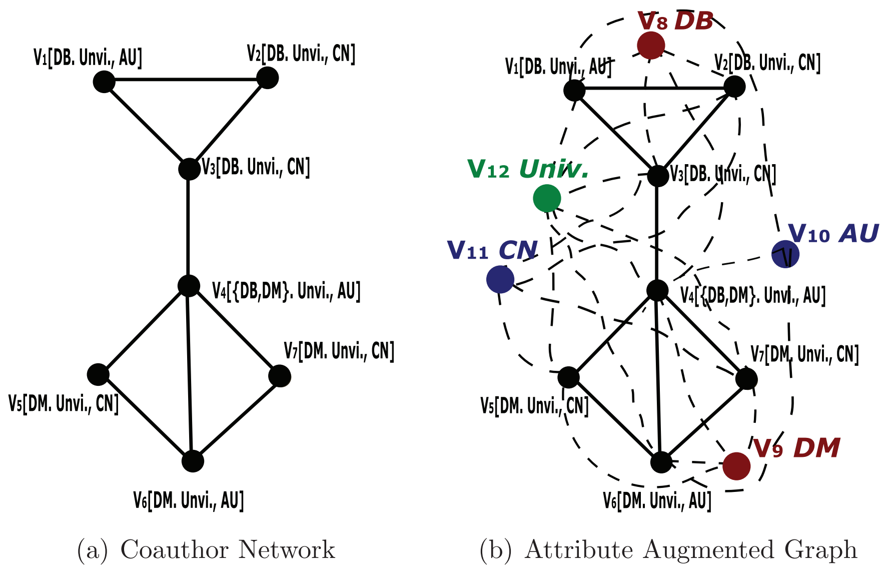


Fig. 3. An example of attribute augmented graph with topics, affiliation and country.

We use $N(v_i)$ to denote the set of structure vertices in v_i's neighborhood, i.e., $N(v_i) = \{v_j | v_j \in V, (v_i, v_j) \in E\}$, and use $N_A(v_i)$ to denote the set of attribute vertices in v_i's neighborhood, i.e., $N_A(v_i) = \{v_{jk} | v_{jk} \in V_a, (v_i, v_{jk}) \in E_a\}$. For any $v_i \in V$, we assume $\sum_{v_j \in N(v_i)} P_A(v_i, v_j) = \sum_{v_{jk} \in N_A(v_i)} P_A(v_i, v_{jk}) = 0.5$, i.e., structure edges take 0.5 transition probability and attribute edges take 0.5 transition probability. Thus the transition matrix P_A is defined as follows. The transition probability from vertex v_i to v_j is

$$p_{v_i,v_j} = \begin{cases} \frac{w(v_i,v_j)}{2\sum_{(v_i,v_k)\in E} w(v_i,v_k)}, & \text{if } (v_i, v_j) \in E \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Similarly, we define the transition probability from vertex v_i to attribute vertex v_{jk} as

$$p_{v_i,v_{jk}} = \begin{cases} \frac{1}{2|\mathcal{A}||A_j(v_i)|}, & \text{if } (v_i, v_{jk}) \in E_a \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

The transition probability from vertex v_{jk} to vertex v_i is

$$p_{v_{jk},v_i} = \begin{cases} \frac{1}{|N(v_{jk})|}, & \text{if } (v_{jk}, v_i) \in E_a \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where $N(v_{jk})$ denotes the set of vertices in v_{jk}'s neighborhood. The transition probability between two attribute vertices v_{ik} and v_{jl} is 0 since there is no edge between them.

$$p_{v_{ik},v_{jl}} = 0, \forall v_{ik}, v_{jl} \in V_a \quad (6)$$

Integrating Eqs. (3)–(6) together, we can get the transition probability matrix P_A on the attribute augmented graph. Following Eq. (2) we can compute the random walk matrix R_A from P_A. $R_A(v_i, v_j)$ is the closeness between vertices v_i and v_j in the attribute augmented graph.

We denote R_A as four submatrices

$$R_A = \begin{bmatrix} Q_{VV} & Q_{VA} \\ Q_{AV} & Q_{AA} \end{bmatrix}$$

where Q_{VV} is an $N \times N$ matrix representing the random walk scores between structure vertices, Q_{VA} is an $N \times N_A$ matrix representing the random walk scores from structure vertices to attribute vertices, Q_{AV} is an $N_A \times N$ matrix representing the random walk scores from attribute vertices to structure vertices, and Q_{AA} is an $N_A \times N_A$ matrix representing the random walk scores between attribute vertices.

We further perform a row normalization of Q_{VV} as

$$\widetilde{Q}_{VV}(v_i, v_j) = \frac{Q_{VV}(v_i, v_j)}{\sum_{v_k \in V} Q_{VV}(v_i, v_k)} \tag{7}$$

With the normalized scores, it is easier for users to determine the cluster connectivity threshold, i.e., not depending on the absolute scores which may vary from graph to graph.

3.4. Combining cells with categorical attributes

Let us get back to the first problem: *how to measure the similarity between two categorical values and determine which categorical values are "adjacent" for merging cells with categorical attributes?* As an example, consider attribute *Conference* in a bibliographic data, with its domain {*VLDB, SIGMOD, ICDE, ICCV, CVPR,* ...}. How can we tell which conferences are more similar and which are less similar? Intuitively, if many authors publish in both *VLDB* and *SIGMOD*, then *VLDB* and *SIGMOD* are similar. Consider the random walk submatrix Q_{AA}. For two different values a_{il}, a_{ir} of the same categorical attribute A_i, the entry $Q_{AA}(v_{il}, v_{ir})$ measures the random walk closeness between these two values. The rationale is, if a_{il} and a_{ir} of attribute A_i are shared by many vertices, they will have a high random walk score, indicating their closeness in the attribute augmented graph. Thus we define the categorical value adjacency based on random walk score. For an attribute value a_{il}, the adjacent values are defined as

$$Adj(a_{il}) = \{a_{ir} | TopK_{r=1, r\neq l}^{n_i}(Q_{AA}(v_{il}, v_{ir}))\} \tag{8}$$

The adjacent values of a categorical attribute value a_{il} is the set $\{a_{ir}\}$ with the top-K largest random walk score $Q_{AA}(v_{il}, v_{ir})$. Then in the cell-based subspace clustering approach, we can combine two k-dimensional cells, if they share common values in $(k-1)$ dimensions, and on the remaining dimension they have either adjacent numerical values or adjacent categorical values as defined above.

3.5. Handling multi-valued attributes

Let us consider the second problem we listed: *if a vertex has multiple values on one attribute, which cell should the vertex be assigned to in the formed grid?* Consider vertex v_4 in Fig. 1(a), which has values *DB* and *DM* on attribute *Topic*. Should v_4 be assigned to the "*DB*" cluster or the "*DM*" cluster? If we consider v_4's neighborhood, we can find v_4 only has one collaborator v_3 working on *DB*, but has three collaborators v_5, v_6, v_7 working on *DM*. This indicates *DM* is more important to v_4 than *DB* (in this example we assume v_4 has the same collaboration strength with each of his collaborators, as concrete edge weights are not given in Fig. 1(a). In our algorithm, we actually consider the edge weights). To distinguish the multi-values, we consider the random walk submatrix Q_{VA}. An entry $Q_{VA}(v_i, v_{jk})$ measures the closeness between vertex v_i and attribute vertex v_{jk}. The rationale is, besides vertex v_i itself has attribute value a_{jk}, if many of v_i's neighbors also have attribute value a_{jk}, v_i and v_{jk} will have a high random walk score, indicating their closeness in the attribute augmented graph. Thus we will reassign the attribute value a_{jk} to v_i which has the maximum closeness to v_i, i.e.,

$$A_j(v_i) = a_{jk}, k = \arg\max_{r=1}^{n_j} Q_{VA}(v_i, v_{jr}) \tag{9}$$

Finally, each vertex is associated with an attribute vector containing a single value in each attribute.

Remark 3.2. Assume that for a co-author network in Fig. 1(a), one author v_4 has the same number of collaborators respectively working on *DB* and *DM* in its neighborhood. But, for v_4, its closeness value to attributes "DB" and "DM" could still be different. This is because even with the same number of "DB" and "DM" authors, the neighborhood structure of "DB" and "DM" clusters can still differ. Then, in Eq. (1), the closeness values are calculated based on the graph structure and attributes vectors of v_4's neighbors, which would be different. Actually, in real applications, the tie closeness will seldomly happen, which helps to choose a unqiue attribute value with the highest closeness score.

4. Our algorithm SCMAG

Our graph subspace clustering framework is similar to that of ENCLUS described in Section 3.2. In the following, we will first define the criteria for interesting subspace with good clustering tendency and find such subspaces. As each attribute A_i has n_i values, for a subspace $\{A_1, \ldots, A_k\}$, the k-dimensional space is partitioned to form a grid. We will identify cells in the grid with high coverage and high connectivity. Adjacent qualified cells will be merged to form a maximal cluster in the subspace.

4.1. Criteria of subspace clustering

We follow the entropy-based subspace clustering method ENCLUS [5] to find interesting subspaces. The subspace entropy is defined in the Definition 3.2.

Remark 4.1. Note that the definition of subspace entropy assumes that each attribute of an object only has one value. Thus, ENCLUS cannot be directly applied to compute the subspace entropy due to the multi-values in a single attribute. On the other hand, our method SCMAG can handle such graphs. The variant methods of ENCLUS for multi-valued attribute graphs will be discussed in Section 5.1.

In addition, we want the attributes of a subspace to be correlated. If the attributes are independent of each other, the subspace does not give more information than looking at each attribute independently. We measure the correlation of a subspace $\mathcal{S}$ using mutual information between all individual dimensions of the subspace as below.

$$I(\{A_1,\ldots,A_k\}) = \sum_{i=1}^{k} H(A_i) - H(A_1,\ldots,A_k)$$

We consider a subspace $\mathcal{S} = \{A_1,\ldots,A_k\}$ as an interesting subspace, if $\mathcal{S}$ is more strongly correlated than any of its subsets $\mathcal{S}' \subseteq \mathcal{S}$. To measure the increase in correlation of a subspace, we define the *interest* of a subspace.

Definition 4.1 (*Subspace Interest*). Given a set of attributes $\mathcal{S} = \{A_1,\ldots,A_k\} \subseteq \mathcal{A}$, the subspace interest of $\mathcal{S}$ is defined as the minimum increase in correlation of $\mathcal{S}$ over its $(k-1)$-dimensional subsets.

$$interest(A_1,\ldots,A_k) = I(\{A_1,\ldots,A_k\}) - \max_i I(\{A_1,\ldots,A_k\} - \{A_i\})$$

Given a maximum entropy threshold e_{max} and a minimum interest threshold i_{min}, we want to find all subspaces $\mathcal{S} \subseteq \mathcal{A}$ such that $H(\mathcal{S}) \leqslant e_{max}$ and $interest(\mathcal{S}) \geqslant i_{min}$. e_{max} guarantees the low entropy of generated subspaces, and i_{min} contributes to throw out subspaces with low increased correlation.

Under a subspace $\mathcal{S} = \{A_1,\ldots,A_k\}$, the space is partitioned to form a grid. We identify cells with high coverage and connectivity, according to the following definition.

Definition 4.2 (*Coverage and Connectivity*). Given a cell u in a subspace, the coverage of u is measured by the number of vertices in u, i.e., $V(u) = |u|$. The connectivity of u is measured by the sum of random walk scores of all pairs of vertices, divided by the cell size

$$D(u) = \frac{\sum_{v_i,v_j\in u} \widetilde{Q}_{VV}(v_i, v_j)}{|u|} \tag{10}$$

Given a minimum coverage threshold v_{min} and a minimum connectivity threshold d_{min}, we only consider cells such that $V(u) \geqslant v_{min}$ and $D(u) \geqslant d_{min}$. We will discuss the parameter setting in Section 4.5.

4.2. Properties

From the above definitions of subspace clustering criteria, we can derive two lemmas as below.

Lemma 4.1. *If a k-dimensional subspace $\{A_1,\ldots,A_k\}$ has entropy below the threshold e_{max} and there exists at least one high coverage cluster C, then the property is established in any $(k-1)$-dimensional projections of this space.*

This property has been proved in [2,5]. From Lemma 4.1, we can use the bottom-up search strategy to find interesting subspaces level by level.

Lemma 4.2. *Suppose two cells u_1, u_2 both satisfy the d_{min} and v_{min} thresholds. When merging u_1 and u_2 to form a cluster $C = u_1 \cup u_2$, C also satisfies $V(C) \geqslant v_{min}$ and $D(C) \geqslant d_{min}$.*

Proof 4.1. $V(C) = V(u_1) + V(u_2) \geqslant v_{min}$,

$$D(C) = \frac{\sum_{v_i,v_j\in u_1} \widetilde{Q}_{VV}(v_i, v_j) + \sum_{v_i,v_j\in u_2} \widetilde{Q}_{VV}(v_i, v_j)}{|u_1| + |u_2|} + \frac{\sum_{v_i\in u_1,v_j\in u_2} \widetilde{Q}_{VV}(v_i, v_j) + \sum_{v_i\in u_2,v_j\in u_1} \widetilde{Q}_{VV}(v_i, v_j)}{|u_1| + |u_2|}$$

$$\geqslant \frac{\sum_{v_i,v_j\in u_1} \widetilde{Q}_{VV}(v_i, v_j) + \sum_{v_i,v_j\in u_2} \widetilde{Q}_{VV}(v_i, v_j)}{|u_1| + |u_2|} \geqslant \frac{|u_1| \times d_{min} + |u_2| \times d_{min}}{|u_1| + |u_2|} = d_{min} \qquad \square$$

4.3. Mining subspace clusters

Algorithm 1. SCMAG

```
Input: $G, e_{max}, i_{min}, v_{min}$, and $d_{min}$.
Output: all clusters satisfying coverage and connectivity requirements in interesting subspaces.

 1:  Calculate the unified random walk matrix $R_A$
 2:  Row normalize the submatrix $Q_{VV}$ of $R_A$ as $\widetilde{Q}_{VV}$
 3:  Assign each vertex $v_i$ with new attribute value in multi-valued attributes according to Eq. (9)
 4:  Let $k = 1$ and $S_k$ be all one-dimensional subspaces
 5:  for each subspace $s \in S_k$ do
 6:    if $H(s) \leqslant e_{max}$, and exists a cell $u$ in $s$ s.t. $V(u) \geqslant v_{min}$ then
 7:      $CS_k = CS_k \cup \{s\}$
 8:      if $interest(s) \geqslant i_{min}$, and exists a cell $u$ in $s$ s.t. $D(u) \geqslant d_{min}$ then
 9:        $Result = Result \cup \mathrm{Find-Clusters}(s)$
10:  $S_{k+1} = \mathrm{Generate-Candidate}(CS_k)$
11:  if $S_{k+1} = \emptyset$, then go to step 13
12:  $k = k + 1$, go to step 5
13:  Output clusters in $Result$
```

According to Lemma 4.1, we will use a bottom-up search strategy, similar to the APRIORI style approach, to find subspaces satisfying e_{max} and i_{min} thresholds. We will start with 1-dimensional subspaces and join two k-dimensional subspaces which share a common prior $(k-1)$-dimensional subspace to form a $(k+1)$-dimensional subspace candidate by a procedure $\mathrm{Generate-Candidate}$. If a subspace violates e_{max} requirement, we do not need to further grow it. Once interesting subspaces are found, a procedure $\mathrm{Find-Clusters}$ is invoked to find subspace clusters. Algorithm 1 shows the subspace clustering algorithm SCMAG.

Algorithm 2 lists the procedure $\mathrm{Find-Clusters}$ for finding subspace clusters. We first identify cells with high coverage and connectivity in a subspace s. Then we perform a depth-first search to merge adjacent cells to form a maximal cluster. $\mathrm{DFS-Find-Clusters}$ is a recursive algorithm which considers each attribute value a_{jr_j} of a cell u in the k-dimensional space and tries to combine u with other cells having adjacent values to a_{jr_j} on attribute A_j and identical values on the remaining attributes. Finally all cells with the same label $cluster\_num$ forms a maximal cluster in the subspace.

Algorithm 2. Find-Clusters

```
Input: All cells in subspace $s = \{A_1, \ldots, A_k\}$, $v_{min}$, and $d_{min}$.
Output: all clusters in $s$.

 1:  Initialize $cluster\_num = 1$
 2:  for each cell $u$ in $s$ do
 3:    if $V(u) < v_{min}$ or $D(u) < d_{min}$ then
 4:      Remove $u$ from $s$
 5:  for each cell $u$ in $s$ do
 6:    if $u.clu\_id$ is undefined then
 7:      $\mathrm{DFS-Find-Clusters}(u, cluster\_num)$
 8:      $cluster\_num = cluster\_num + 1$
 9:  Procedure $\mathrm{DFS-Find-Clusters}$
     Input: starting cell $u = [a_{1r_1}, \ldots, a_{kr_k}], cluter\_num$.
     Output: a maximal cluster.
10:  Set $u.clu\_id = cluter\_num$
11:    for $j = 1 : k$ do
12:      for each $a_{jt} \in Adj(a_{jr_j})$ do
13:        $u_t = [a_{1r_1}, \ldots, a_{jt}, \ldots, a_{kr_k}]$
14:        if $u_t.clu\_id$ is undefined then
15:          $\mathrm{DFS-Find-Clusters}(u_t, cluster\_num)$
```

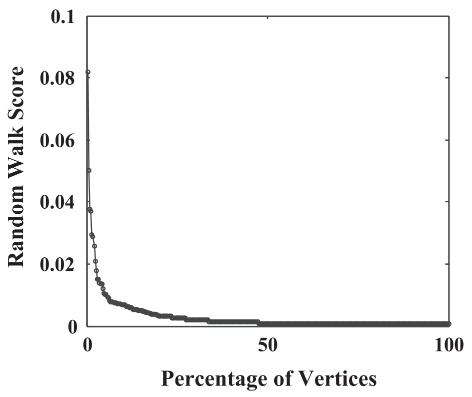


Fig. 4. Random walk scores.

4.4. Computing cell connectivity efficiently

We impose a connectivity requirement on a cell u, i.e., $D(u) = \frac{\sum_{v_i,v_j \in u} \widetilde{Q}_{VV}(v_i,v_j)}{|u|} \geqslant d_{min}$. It takes $O(|u|^2)$ to calculate $D(u)$ for a cell u. To speed up the connectivity calculation, we design an index and a connectivity upper bound to prune cells with low connectivity.

The index is based on the following observation: for a vertex $v \in V$, there are only a small number of vertices with high closeness to v, but a majority of vertices have very low closeness to v. Fig. 4 shows the random walk scores of a vertex to all other vertices in a graph based on a bibliographic data we used in experiments. We can observe that most of the vertices have a very low random walk score close to 0. Therefore, we maintain two vertex sets for each vertex – one set of vertices with high closeness, denoted as *HC*, and the other set of vertices with low closeness, denoted as *LC*. We will start computing $D(u)$ from the *HC* sets of vertices in u and see if we can stop early and tell whether $D(u) \geqslant d_{min}$ or not.

We build the *HC* and *LC* sets for each vertex in *V* as follows. Consider the random walk submatrix $\widetilde{Q}_{VV}$. For a vertex $v_i \in V$, we sort all entries $\widetilde{Q}_{VV}(v_i, v_j), \forall v_j \in V$ in the decreasing order. We insert the sorted entries $\widetilde{Q}_{VV}(v_i, v_j)$ into $v_i.HC$ one by one until $\sum_{v_j \in v_i.HC} \widetilde{Q}_{VV}(v_i, v_j) \geqslant \delta$, where $\delta \in [0, 1)$ is a fixed threshold for all vertices. The remaining entries are inserted into $v_i.LC$. Table 2 shows the *HC* and *LC* sets for vertices in a graph under the setting $\delta = 0.75$. As we can see, the *HC* set usually contains a small number of vertices with high closeness, and *LC* contains a large number of vertices with low closeness.

For a cell u, we calculate $D(u)$ in two stages. In the first stage we calculate the closeness entries in the *HC* sets only, as an estimation of the true $D(u)$ value:

$$D_{HC}(u) = \frac{\sum_{v_i \in u} \sum_{v_j \in u, v_j \in v_i.HC} \widetilde{Q}_{VV}(v_i, v_j)}{|u|} \tag{11}$$

If $D_{HC}(u) \geqslant d_{min}$, we can stop early and conclude $D(u) \geqslant d_{min}$. If $D_{HC}(u) < d_{min}$, we move to the second stage to compute $D_{LC}(u)$ as

$$D_{LC}(u) = \frac{\sum_{v_i \in u} \sum_{v_j \in u, v_j \in v_i.LC} \widetilde{Q}_{VV}(v_i, v_j)}{|u|} \tag{12}$$

Finally we have $D(u) = D_{HC}(u) + D_{LC}(u)$. In addition, we can also derive an upper bound of $D(u)$ to prune those cells with $D(u) < d_{min}$.

Lemma 4.3. *Consider a cell $u = \{v_1, \ldots, v_k\}$. We compute the cell connectivity $D(u)$ in two stages. In the first stage, we calculate $D_{HC}(u)$ according to Eq. (11) by iterating over each vertex in u. During this stage, if $D_{HC}(u) \geqslant d_{min}$, we can stop and conclude $D(u) \geqslant D_{HC}(u) \geqslant d_{min}$. In the second stage, we calculate $D_{LC}(u)$ according to Eq. (12) by iterating over each vertex in u.*

In the first stage, assume we have iterated over vertices $\{v_1, \ldots, v_l\}$. Let $w = \sum_{i=1}^{l} \sum_{v_j \in u, v_j \in v_i.HC} \widetilde{Q}_{VV}(v_i, v_j)$, then we can have an upper bound as $\widehat{D}_1(u) = \frac{w+k-\delta l}{|u|}$.

In the second stage, assume we have iterated over vertices $\{v_1, \ldots, v_l\}$. Let $w' = \sum_{i=1}^{k} \sum_{v_j \in u, v_j \in v_i.HC} \widetilde{Q}_{VV}(v_i, v_j) + \sum_{i=1}^{l} \sum_{v_j \in u, v_j \in v_i.LC} \widetilde{Q}_{VV}(v_i, v_j)$, then we can have an upper bound as $\widehat{D}_2(u) = \frac{w'+(1-\delta)(k-l)}{|u|}$.

If either $\widehat{D}_1(u) < d_{min}$ or $\widehat{D}_2(u) < d_{min}$, then we can immediately stop and conclude $D(u) < d_{min}$.

Proof 4.2. The foundation to derive the upper bounds is that, $\widetilde{Q}_{VV}$ is a row normalized matrix. That is, $\forall v_i \in V$, the following holds:

$$\sum_{v_j \in v_i.HC} \widetilde{Q}_{VV}(v_i, v_j) + \sum_{v_{j'} \in v_i.LC} \widetilde{Q}_{VV}(v_i, v_{j'}) = 1 \tag{13}$$

Table 2
Vertices sorted in HC and LC, $\delta = 0.75$.

Vertex	HC set	LC set
v_1	$(v_5, 0.5), (v_2, 0.28)$	$(v_3, 0.01), \ldots, (v_n, 0.001)$
v_2	$(v_1, 0.3), (v_3, 0.25), (v_9, 0.2)$	$(v_6, 0.03), \ldots, (v_n, 0.003)$
...	...	...
v_n	$(v_{10}, 0.5), (v_8, 0.35)$	$(v_7, 0.01), \ldots, (v_1, 0.002)$

According to the definitions of HC and LC sets, we have:

$$\sum_{v_j \in v_i.HC} \widetilde{Q}_{VV}(v_i, v_j) \geqslant \delta \tag{14}$$

$$\sum_{v_j \in v_i.LC} \widetilde{Q}_{VV}(v_i, v_j) \leqslant 1 - \delta \tag{15}$$

Based on these properties we can derive:

$$\begin{aligned} D(u) &= \frac{\sum_{i=1}^{k}\sum_{j=1}^{k}\widetilde{Q}_{VV}(v_i, v_j)}{|u|} = \frac{\sum_{i=1}^{l}\sum_{v_j\in u, v_j\in v_i.HC}\widetilde{Q}_{VV}(v_i, v_j)}{|u|} + \frac{\sum_{i=1}^{l}\sum_{v_j\in u, v_j\in v_i.LC}\widetilde{Q}_{VV}(v_i, v_j)}{|u|} + \frac{\sum_{i=l+1}^{k}\sum_{j=1}^{k}\widetilde{Q}_{VV}(v_i, v_j)}{|u|} \\ &\leqslant \frac{w + (1-\delta)l + (k-l)}{|u|} = \frac{w + k - \delta l}{|u|} = \widehat{D}_1(u) \end{aligned}$$

The inequality holds because $\sum_{i=1}^{l}\sum_{v_j\in u, v_j\in v_i.LC}\widetilde{Q}_{VV}(v_i, v_j) \leqslant (1-\delta)l$ according to Eq. (15), and $\sum_{i=l+1}^{k}\sum_{j=1}^{k}\widetilde{Q}_{VV}(v_i, v_j) \leqslant k - l$ according to Eq. (13). Thus we can derive the upper bound $\widehat{D}_1(u)$.

Similarly, to prove $\widehat{D}_2(u)$ we have

$$\begin{aligned} D(u) &= \frac{\sum_{i=1}^{k}\sum_{j=1}^{k}\widetilde{Q}_{VV}(v_i, v_j)}{|u|} = \frac{\sum_{i=1}^{l}\sum_{j=1}^{k}\widetilde{Q}_{VV}(v_i, v_j)}{|u|} + \frac{\sum_{i=l+1}^{k}\sum_{v_j\in u, v_j\in v_i.HC}\widetilde{Q}_{VV}(v_i, v_j)}{|u|} + \frac{\sum_{i=l+1}^{k}\sum_{v_j\in u, v_j\in v_i.LC}\widetilde{Q}_{VV}(v_i, v_j)}{|u|} \\ &\leqslant \frac{w' + (1-\delta)(k-l)}{|u|} = \widehat{D}_2(u) \end{aligned}$$

The inequality holds as $\sum_{i=l+1}^{k}\sum_{v_j\in u, v_j\in v_i.LC}\widetilde{Q}_{VV}(v_i, v_j) \leqslant (1-\delta)(k-l)$ according to Eq. (15). Thus we can derive the upper bound $\widehat{D}_2(u)$. □

Algorithm 3. Calculate-Connectivity

```
Input: A cell $u = \{v_1, \ldots, v_k\}$ which contains $k$ vertices, minimum cell connectivity $d_{min}$, a fixed threshold $\delta$.
Output: $D(u) \geqslant d_{min}$ or not.

1:  initialize $w = 0$
    /* first stage */
2:  for $l = 1 : k$ do
3:    for $v_j \in u, v_j \in v_l.HC$ do
4:      $w = w + \widetilde{Q}_{VV}(v_l, v_j)$
5:    if $w/|u| \geqslant d_{min}$ then
6:      return $D(u) \geqslant d_{min}$
7:    if $(w + k - \delta l)/|u| < d_{min}$ then
8:      return $D(u) < d_{min}$
    /* second stage */
9:  for $l = 1 : k$ do
10:   for $v_j \in u, v_j \in v_l.LC$ do
11:     $w = w + \widetilde{Q}_{VV}(v_l, v_j)$
12:   if $w/|u| \geqslant d_{min}$ then
13:     return $D(u) \geqslant d_{min}$
14:   if $(w + (1-\delta)(k-l))/|u| < d_{min}$ then
15:     return $D(u) < d_{min}$
```

We design an efficient algorithm Calculate-Connectivity in Algorithm 3 for calculating connectivity based on Lemma 4.3.

4.5. Parameters setting

SCMAG has four parameters $e_{max}, i_{min}, v_{min}$ and d_{min}. For a k-dimensional subspace $\mathcal{S} = \{A_1, \ldots, A_k\} \subseteq \mathcal{A}$, the subspace entropy $H(A_1, \ldots, A_k) \in [0, k\log_2 p]$ holds, if we assume $|Dom(A_i)| = p$ for each attribute $A_i \in \mathcal{S}$. $k\log_2 p$ is reached when the k attributes are independent of each other. Thus the maximum entropy threshold e_{max} can be determined within the range $[0, k\log_2 p]$. In practice, for detecting good clusters in $k(2 \leqslant k \leq 6)$-dimensional subspace with $p = 20$, we find that $e_{max} = 10$ always yields a good performance. Similarly, for a k-dimensional subspace $\mathcal{S} = \{A_1, \ldots, A_k\} \subseteq \mathcal{A}$, we can derive $interest(A_1, \ldots, A_k) \in [0, \log_2 p]$. Thus the minimum interest threshold i_{min} can be determined within $[0, \log_2 p]$. $i_{min} \leqslant 1$ is a good choice, which can avoid producing too few clusters.

The minimum cell coverage threshold v_{min} should not be set too large, since only gigantic clusters will be generated in that case. v_{min} depends on the graph size and user interests. In our experiments, for a graph with $|V| \geqslant 10{,}000$, we usually choose v_{min} no less than 100. For a cell $u, D(u) \in [0, 1]$ holds. Thus d_{min} can be determined within $[0, 1]$. d_{min} should not be set too low, and $d_{min} \in [0.2, 0.6]$ is highly recommended to find densely connected clusters.

4.6. Time complexity analysis

Our algorithm has two major components: random walk and subspace clustering. The time complexity of calculating random walk is $O(L(|V| + |V_a|)^{\omega})$, where $\omega < 2.376$ is the fast matrix product exponent [7], and $|V|$ is the number of structure vertices and $|V_a| = \sum_{i=1}^{d} n_i$ is the number of attribute vertices. Under an attribute subspace, the time complexity of calculating subspace entropy is $O(|V|d)$, where d is the attribute number, and the time complexity of calculating cell connectivity is $O(|V|^2)$. The total number of qualified subspaces in worst would be 2^d, then the time complexity of subspace clustering is $O((|V|d + |V|^2)2^d)$. As a result, the total time complexity of our algorithm is $O(L(|V| + |V_a|)^{2.376} + (|V|d + |V|^2)2^d)$. In real application, the number of qualified interesting subspaces can be controlled by our parameters e.g. e_{max} and d_{min}, and would be far less than the worst case. We would test the number of qualified subspaces in our experiments.

5. Experimental study

In this section, we conducted extensive experiments to evaluate the performance of our algorithm on the IMDB movie dataset and a bibliographic dataset. Moreover, we presented some interesting case studies. Our experiments were run on a Dell PowerEdge R900 server with 2.67 GHz six-core CPU and 128 GB main memory running Windows Server 2008. All algorithms were implemented in C++, except that random walk was implemented in MATLAB.

5.1. Comparison methods

We evaluate the clustering quality and efficiency of our method. We compare with ENCLUS [5], which does not consider structural connectivity between instances in clustering. We develop two variations of ENCLUS, which only differ in handling multi-valued attributes as described below.

- Single-ENCLUS For instances having multi-values in an attribute, Single-ENCLUS randomly selects one of the values and ignores the rest.
- Multi-ENCLUS For instances having multi-values in an attribute, Multi-ENCLUS creates multiple single-valued instances for each of the multi-values.

We also compare with a closely related method GAMer [12,13] which combines subspace clustering with dense subgraph mining. Since GAMer outperforms CoPaM [19] in terms of cluster quality and efficiency, we do not compare with CoPaM in the experiment. The GAMer program is publicly available at http://dme.rwth-aachen.de/en/gamer. We first transform multi-valued attributed data to binary vectors. GAMer has several parameters and we follow the recommended parameter settings in its program. n_{min} is the minimum cluster size (the same as our parameter v_{min}), and γ_{min} is the minimum density of a cluster, which is similar as our parameter d_{min} but different in definition. A non-empty cluster C fulfills the density requirement if $\frac{\min_{v \in C}\{|\{(v,u)|(v,u) \in E, u \in C\}|\}}{|C|-1} \geqslant \gamma_{min}$. s_{min} is the minimum subspace dimension, which should be at least 2. Another parameter maximal width w should be set to 0 to handle categorical attributes, i.e., only vertices with the same categorical values will fall into the same cell. The ranges of parameters n_{min} and γ_{min} are listed in Table 3, and other parameters are set to the default values in Table 4. We test different parameter value combinations in the experiments.

Table 3
Parameter ranges of n_{min} and γ_{min} in GAMer.

Parameter	Range
n_{min}	10, 20, 50, 100, 200, 300, 400, 500, 600, 700
γ_{min}	0.10, 0.20, 0.30, 0.40, 0.50, 0.60

Table 4
Default parameter settings in GAMer.

Parameter	s_{min}	w	a	b	c	r_{obj}	r_{dim}
Default value	2	0	0	1	0	0.5	0.5

We also compare our method with the full space attributed graph clustering SA-Cluster [6] in Section 5.7.

5.2. Evaluation measures

We use density and entropy defined below to evaluate the quality of clusters. Density measures the intra-cluster edge weights normalized by cluster size, thus the larger the better. Entropy measures the attribute value distribution within a subspace cluster, thus the lower the better.

$$density(\{C_i\}_{i=1}^{R}) = \frac{1}{R}\sum_{i=1}^{R}\frac{\sum_{u,v\in C_i} w(u,v)}{|\{v|v\in C_i\}|} \tag{16}$$

$$entropy(\{C_i\}_{i=1}^{R}) = \frac{1}{R}\sum_{i=1}^{R}\frac{1}{|\mathcal{A}_i|}\sum_{A\in\mathcal{A}_i} entropy(A,C_i) \tag{17}$$

where $entropy(A,C_i) = -\sum_{j=1}^{|Dom(A)|} p_{ij}\log_2 p_{ij}$, and $p_{ij} = \frac{\sum_{v\in C_i, a_j\in A(v)}\frac{1}{|A(v)|}}{|\{v|v\in C_i\}|}$, i.e., the percentage of vertices in cluster C_i having value a_j on attribute A. In the above definitions, density only depends on edge connectivity in clusters, which is not relevant with attribute space. Entory is defined in subspace. For a cluster $C_i, \mathcal{A}_i$ is the attribute subspace of C_i, where $\mathcal{A}_i \subset \mathcal{A}$.

5.3. IMDB

5.3.1. Dataset description

We use IMDB data[1] to build a multi-valued attributed graph, where a vertex represents a movie and an edge exists between two movies if they share one or more actors. The edge weight is the number of common actors. The multi-valued attributed graph contains 17,506 movies and 609,996 edges. A vertex is associated with 8 attributes which are described in Table 5. The numerical attributes *Actor Number* and *Voter Number* are discretized using equal-frequency binning, and *Average Rating* and *Variance of Rating* are discretized using equal-width binning. The release year of all movies is between 1990 and 2000. The other three attributes are categorical. A movie may have multi-values in *Genres*, e.g., the genres of movie "The Shawshank Redemption" are *crime* and *drama*.

Unless otherwise specified, we use the default parameter values shown in Table 6, and set the indexing threshold $\delta = 0.8$. We test three different d_{min} values of 0.30, 0.15, and 0.0.

5.3.2. Cluster quality comparison

In the first experiment, we compare the cluster quality of our method SCMAG and ENCLUS based methods by varying the parameter e_{max} – maximum allowed subspace entropy.

Fig. 5(a) shows the entropy. The entropy by our method SCMAG with $d_{min} = 0.30$ and 0.15 consistently outperforms the other methods. This shows when a connectivity requirement is enforced, high-quality clusters are discovered from subspaces which have not only a dense structure but also similar attribute values. This phenomenon can be interpreted as follows: if a set of movies share many actors, i.e., dense connectivity, they also have similar attribute values. In contrast, the entropy of Single-ENCLUS and Multi-ENCLUS is much higher. When $d_{min} = 0.0$, the entropy of SCMAG increases and is close to that of Single-ENCLUS. Moveover, a general trend we observe is that entropy decreases for all methods when the parameter e_{max} increases.

Fig. 5(b) shows the density. SCMAG with $d_{min} = 0.30$ achieves the highest density and outperforms the other methods by a large margin. This is natural as we enforce a stricter connectivity requirement with a larger d_{min} value, the resultant clusters have a larger density. In addition, our method with $d_{min} = 0.30$ and 0.15 shows a stable density when e_{max} increases. In contrast, the other three methods have a decreasingly small density when e_{max} increases.

In the second experiment, we compare the cluster quality of different methods by varying the parameter v_{min} – minimum cell coverage. As we can see from Fig. 6(a), SCMAG with $d_{min} = 0.30$ has the smallest entropy around 0.5–0.7. It also shows a decreasing trend when v_{min} increases. The other baseline methods have a much larger entropy. In Fig. 6(b), SCMAG with $d_{min} = 0.30$ achieves the largest density and outperforms the other methods by a large margin.

From these two experiments, we can conclude SCMAG with $d_{min} = 0.30$ performs the best on both entropy and density. Multi-ENCLUS performs the worst on both criteria.

[1] http://www.imdb.com.

Table 5
IMDB attributes.

ID	Name	Type	Domain
1	Actor number	Numerical	10
2	Voter number	Numerical	20
3	Average rating	Numerical	20
4	Variance of rating	Numerical	20
5	Year	Numerical	11
6	Genres	Categorical, multi-value	29
7	Language	Categorical, single-value	135
8	Sound-mix	Categorical, single-value	33

Table 6
Default parameter values: IMDB.

Parameter	e_{max}	i_{min}	v_{min}	d_{min}
Default value	10	0.10	150	0.30

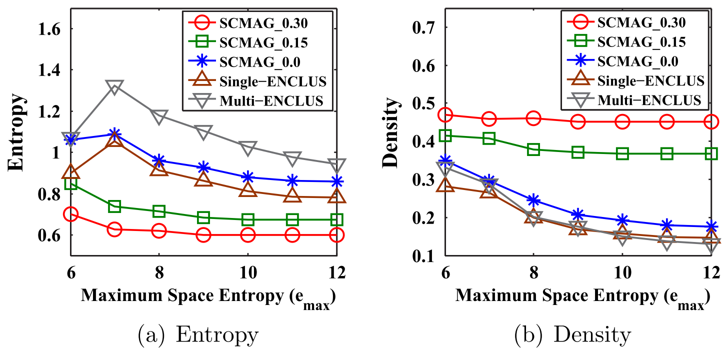


Fig. 5. Cluster quality varying e_{max}: IMDB.

5.3.3. GAMer results

Under the default parameter settings in Table 4, we test GAMer on IMDB data by varying different value combinations of n_{min} and γ_{min} in Table 3. However, GAMer fails to return any meaningful results. The outcomes of all the runs can be summarized as two cases: "timeout" and "no cluster found". "timeout" means GAMer cannot finish in 5 h, and "no cluster found" means GAMer finishes with no cluster found. Fig. 7(a) shows the results under different n_{min} and γ_{min} values, where white cells represent "no cluster found" and gray cells represent "timeout".

We can observe, for example, when $\gamma_{min} = 0.6$ and $n_{min} \geqslant 200$, no cluster can be found, as there is no cluster satisfying the γ_{min} and n_{min} requirements. When $\gamma_{min} = 0.1$ and $10 \leqslant n_{min} \leqslant 700$, GAMer becomes timeout. This is because the search space is exponential in terms of the number of vertices in a graph, GAMer takes extremely long run time and a lot of memory, especially when the pruning is not effective.

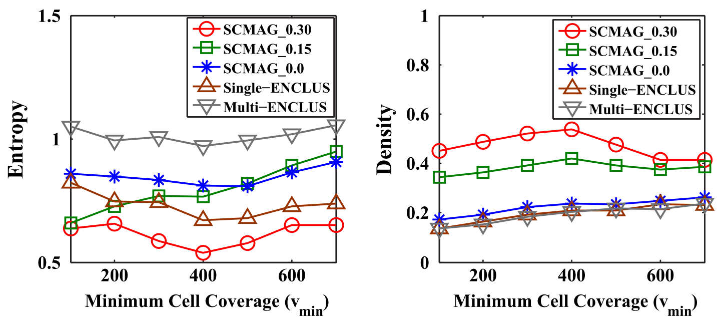
5.3.4. Case study

We present some interesting subspace clusters found by SCMAG with $d_{min} = 0.30$ on IMDB.

We first study a two-dimensional subspace {*Actor Number, Genres*}, under which we only found two clusters shown in Table 7. Cluster 1 contains 2380 drama movies with actor number in $[25, 4185]$, and cluster 2 contains 167 adult movies with actor number in $[5, 6]$. These clusters quite make sense as drama movies usually have a big cast, while adult movies usually have a smaller cast.

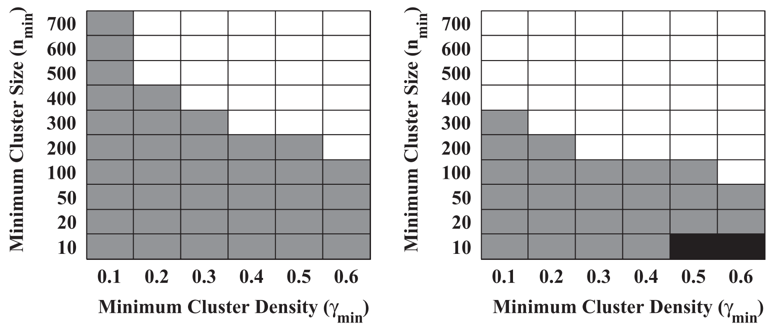
We study another subspace {*Genres, Language*}, under which we found a total of 14 clusters. Table 8 shows the cluster list. We have the following interesting observations.

- Some typical clusters are from a combination of drama or comedy movies with one of the languages in English, German, Spanish, and French (clusters 1–4, 6–9).
- Interestingly, cluster No. 5 for Hindi drama movies is discovered with a high density value of 0.652, showing that these movies share many common actors. It is well known that India has the Bollywood just like America has the Hollywood, which has produced many popular movies.



(a) Entropy (b) Density

Fig. 6. Cluster quality varying v_{min}: IMDB.



(a) IMDB Graph (b) Bibliographic Graph

Fig. 7. GAMer results. White cell: *no cluster found*, gray cell: *timeout*, black cell: *out of memory*.

Table 7
Clusters in subspace {Actor Number, Genres} on IMDB graph.

Cluster	Actor number	Genres	Size	Density
1	[25, 4185]	Drama	2380	0.433
2	[5, 6]	Adult	167	0.316

Table 8
Clusters in subspace {Genres, Language} on IMDB graph.

Cluster	Genres	Language	Size	Density
1	Drama	English	4857	0.554
2	Drama	German	300	0.276
3	Drama	Spanish	387	0.523
4	Drama	French	459	0.409
5	Drama	Hindi	219	0.652
6	Comedy	English	2292	0.377
7	Comedy	German	298	0.303
8	Comedy	Spanish	160	0.416
9	Comedy	French	236	0.307
10	Comedy	Cantonese	189	0.497
11	Action	Cantonese	215	0.410
12	Music/documentary	English	970	0.281
13	Adult	English	525	0.940
14	Animation	English	272	0.396

- Cantonese is a dialect spoken in southern China, Hong Kong, etc. Interestingly, we found a cluster of 189 Cantonese comedy movies (No. 10) with a density of 0.497. Another cluster of 215 Cantonese action movies (No. 11) corresponds to the world famous Chinese Kung Fu movies. These clusters reflect that the film industry of Hong Kong had brilliant achievements in 1990–2000, although Cantonese is only a local dialect.
- Cluster No. 12 is formed by combining music and documentary English movies. According to our mechanism to compute the similarity between categorical values, we find "documentary" is close to "biography" and "music". Thus the cells of [Music, English] and [Documentary, English] can be merged. This cluster could not be found by Single-ENCLUS and Multi-

ENCLUS. Five representative movies in this cluster include: *The Beatles Anthology (1995), Metallica: S&M (2000), The Wall: Live in Berlin (1990), The Filth and the Fury (2000), Metallica: Cunning Stunts (1998)*, all of which have genres of documentary and music.

- Interestingly, cluster No. 13 for adult English movies has a very high density value of 0.940, which reflects that most of these adult movies are starred by a common set of actors.

5.4. Bibliographic data

5.4.1. Dataset description

We use a bibliographic dataset collected from Arnetminer[2] to construct a multi-valued attributed graph, where a vertex represents an author and an edge exists between two authors if they have co-authored one or more papers. The edge weight is the number of co-authored papers by two authors. The multi-valued attributed graph contains 80,000 authors and 476,387 edges. A vertex is associated with 12 attributes which are described in Table 9. *Topic* is a categorical multi-valued attribute describing the research topics of authors, and an author can have multiple topics. We partition all venues in this bibliographic dataset into 45 groups according to area. *Venue* describes a list of venue groups an author has published in them. For an author, we retain at most 3 values for attributes *Topic* and *Venue* respectively. The remaining attributes are numerical and are discretized into 10 bins using equal-width binning. These attribute values are computed by Arnetminer algorithms.

Unless otherwise specified, we use the default parameter values in Table 10, and set the indexing threshold $\delta = 0.8$. We test three different d_{min} values of 0.40, 0.20, and 0.0.

5.4.2. Cluster quality comparison

In the first experiment, we compare the cluster quality of different methods by varying e_{max}. Fig. 8(a) shows the entropy. The entropy by SCMAG with $d_{min} = 0.40$ and 0.20 is very low when $e_{max} = 7, 8, 9$, but the entropy increases given a larger e_{max}. In contrast, the entropy of the other three methods shows a decreasing trend with e_{max}. Multi-ENCLUS has the largest entropy in all cases.

Fig. 8(b) shows the density. Again, our method SCMAG with $d_{min} = 0.40$ achieves the highest density and outperforms the other methods by a large margin. This is due to the d_{min} constraint. In addition, SCMAG with $d_{min} = 0.40$ and 0.20 has a stable density despite the increase of e_{max}, while the other three methods show a decreasing trend with the increase of e_{max}. Single-ENCLUS has the lowest density in all cases. These results demonstrate that our method can find dense clusters in subspaces with low entropy.

In the second experiment, we compare the cluster quality of different methods by varying the parameter v_{min}. Fig. 9(a) shows the entropy. We can observe that all methods have a lower entropy when v_{min} increases. This shows subspace clusters with a larger size have more similar attribute values in the bibliographic data. Our method SCMAG with $d_{min} = 0.20$ and 0.0 and Single-ENCLUS score closely and achieve the lowest entropy. Multi-ENCLUS has the largest entropy in all cases. Fig. 9(b) shows the density. All methods have a stable density when v_{min} increases. SCMAG with $d_{min} = 0.40$ achieves the largest density and outperforms the other methods by a large margin.

From these experiments, we can conclude SCMAG with $d_{min} = 0.20$ achieves a good balance between entropy and density. Multi-ENCLUS performs poorly on both criteria.

5.4.3. GAMer results

Under the default parameter settings in Table 4, we also test GAMer on the bibliographic data by varying different value combinations of n_{min} and γ_{min} in Table 3. Again GAMer fails to return any meaningful results. The outcomes of all the runs can be summarized as three cases: "no cluster found", "timeout" and "out of memory" (GAMer throws a *java.lang.OutOfMemoryError* exception). Fig. 7(b) shows the results under different n_{min} and γ_{min} values, where dark cells represent "out of memory", and white and gray cells are similarly defined as in Fig. 7(a).

We can observe, for example, when $n_{min} = 10$ and γ_{min} = 0.5 or 0.6, "out of memory" happens. When $\gamma_{min} = 0.6$ and $n_{min} \geqslant 100$, no cluster can be found, because there is no cluster satisfying the requirements. When $\gamma_{min} = 0.1$ and $10 \leqslant n_{min} \leqslant 300$, GAMer becomes timeout; while for $400 \leqslant n_{min} \leqslant 700$, no cluster can be found. In summary, GAMer cannot find any meaningful clusters on the bibliographic data within a reasonable time.

5.4.4. Case study

We present some interesting subspace clusters found by SCMAG with $d_{min} = 0.40$ on the bibliographic data.

We first examine a two-dimensional subspace {*Topic, Venue*} under which we found 31 clusters. It is interesting to discover the combination of *Topic* and *Venue*, as research topics are closely correlated with published venues. This shows the subspace interest criterion is useful in finding correlated subspaces. We pick four clusters as examples and present their topics, representative venues, top 10 most cited authors in Table 11. In particular, cluster 3 combines several similar topics, e.g., *communication complexity, approximation algorithms, lower bounds, perfect graphs*, etc., related to algorithms and theory in the *Topic* dimension. Cluster 3 and cluster 4 could not be found by Single-ENCLUS and Multi-ENCLUS.

[2] http://www.arnetminer.org.

Table 9
Bibliographic data attributes.

ID	Name	Type	Domain
1	Topic	Categorical, multi-value	200
2	Venue	Categorical, multi-value	45
3	Publication	Numerical	10
4	Citation	Numerical	10
5	H-index	Numerical	10
6	G-index	Numerical	10
7	Activity	Numerical	10
8	Uptrend	Numerical	10
9	Newstar score	Numerical	10
10	Longevity	Numerical	10
11	Diversity	Numerical	10
12	Sociability	Numerical	10

Table 10
Default parameter values: bibliographic.

Parameter	e_{max}	i_{min}	v_{min}	d_{min}
Default value	10	0.01	200	0.40

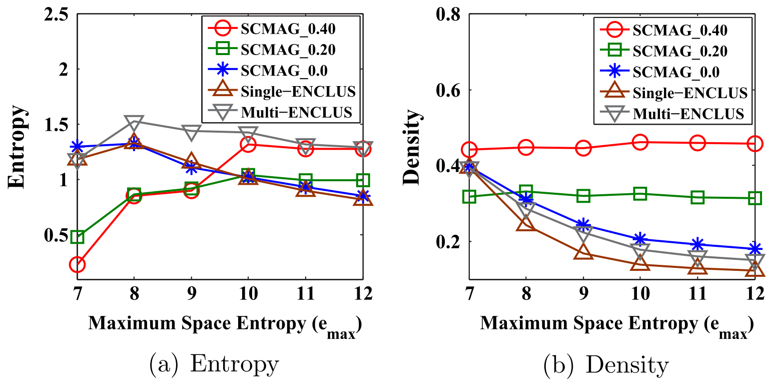


Fig. 8. Cluster quality vs. e_{max}: bibliographic.

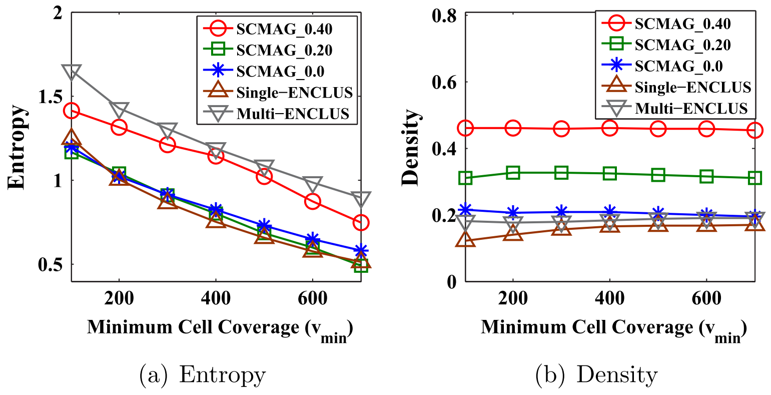


Fig. 9. Cluster quality vs. v_{min}: bibliographic.

We study another subspace {*Activity, Uptrend, Venue*} under which we only found 1 cluster. The cluster has the highest attribute value intervals [1.09, 32.27), [0.75, 23.36) in *Activity* and *Uptrend* respectively. The authors in this cluster are very active, showing a strong uptrend in publication in the area of algorithms and theory (refer to venue information in cluster 3, Table 11). We list 5 authors with the highest uptrend values: *Pierre Fraigniaud, Flavio Chierichetti, David Woodruff, Devavrat Shah, and Phillip B. Gibbons.*

Table 11
Clusters in subspace {Topic, Venue} on bibliographic graph.

Cluster	Topic	Venue	Authors (top-10 citations)
1. Data mining	Data mining	TKDE, KDD, SDM, ICDM, PAKDD, KAIS	Jiawei Han, Sudhir Kumar, Philip S. Yu, Vipin Kumar, George Karypis, Heikki Mannila, Mohammed J. Zaki, Eamonn J. Keogh, Bing Liu, Michael J. Pazzani
2. Database	XML data	SIGMOD, VLDB, ICDE, EDBT, CIKM, Data Eng. Bulletin	Rakesh Agrawal, Jennifer Widom, Christos Faloutsos, Ramakrishnan Srikant, Serge Abiteboul, Miron Livny, Tomasz Imielinski, Johannes Gehrke, Rajeev Rastogi, Dan Suciu
3. Algorithms, theory	Communication complexity, approximation algorithms, finite sets, new designs, convex polygons, planar arrangements, lower bounds, perfect graphs	TCS, TIT, DAM, JCSS, SICOMP, Algorithmica, STOC	Rajeev Motwani, Christos Papadimitriou, Robert E. Tarjan, Prabhakar Raghavan, David R. Karger, Richard M. Karp, Jon M. Kleinberg, Leslie Valiant, Oded Goldreich, Moni Naor
4. Computer vision, pattern recognition	Character recognition, object recognition, two-view motion estimation, face recognition, image analysis	ICIP, ICPR, ICME, PR, CVPR, TPAMI, PRL, ICDAR	Anil K. Jain, Takeo Kanade, Jitendra Malik, Alex Pentland, Andrew Zisserman, Thomas Huang, Andrew Blake, Cordelia Schmid, Guillermo Sapiro, Azriel Rosenfeld

Table 12
Clusters in subspace {Citation, H-index, G-index, Venue} on bibliographic graph.

Cluster 1 database	Cluster 2 software engineering & scientific computing	Cluster 3 hardware & architecture	Cluster 4 algorithms & theory
Rakesh Agrawal	C.A.R. Hoare	A.L. Sangiovanni-Vincentelli	Rajeev Motwani
Hector Garcia-Molina	Leslie Lamport	Sharad Malik	Robert E. Tarjan
Jeffrey D. Ullman	Thomas A. Henzinger	Sartaj K. Sahni	Christos Papadimitriou
Jennifer Widom	Rajeev Alur	Lothar Thiele	Prabhakar Raghavan
Christos Faloutsos	David Harel	Sudhakar M. Reddy	David R. Karger
Jim Gray	Joseph Halpern	Jason Cong	Richard M. Karp
David J. DeWitt	Amir Pnueli	Robert Brayton	Jon M. Kleinberg
Michael Stonebraker	Moshe Vardi	Miodrag Potkonjak	Leslie Valiant
Ramakrishnan Srikant	Edmund Clarke	Massoud Pedram	Oded Goldreich
Serge Abiteboul	Robin Milner	Janak H. Patel	Moni Naor

In the subspace {*Citation, H-index, G-index, Venue*}, SCMAG found four clusters of authors from different research fields. Again the subspace combination of *Citation, H-index* and *G-index* is interesting, as these three attributes are positively correlated – H-index and G-index are computed from citations. Authors in these four clusters all have the highest values in *Citation, H-index* and *G-index* as [696, 51258), [12, 102) and [6, 25) respectively. We list 10 representative authors in each cluster in Table 12.

5.5. Subspace number, cluster number and size

We report the number of attribute subspaces (with valid clusters) found by different algorithms by varying e_{max} in Fig. 10. With the increase of e_{max}, Single-ENCLUS, Multi-ENCLUS and SCMAG with $d_{min} = 0.0$ output an increasingly large number of subspaces. But with a minimum cell connectivity constraint $d_{min} > 0$, SCMAG does not output many subspaces when e_{max} increases, as those subspace clusters which do not meet the connectivity requirement are not output. This result shows our method can better control the subspace cluster quality and output only those clusters with both dense connectivity and high attribute similarity.

We plot the cluster size (x axis) versus cluster number (y axis) in Fig. 11. A point (x, y) in a curve means there are y clusters whose size is above x. On both IMDB and bibliographic graphs we can see the curves of Single-ENCLUS, Multi-ENCLUS and SCMAG with $d_{min} = 0.0$ largely overlap, as they do not have any structural connectivity requirement. We observe the following phenomena: (1) the size of clusters found by SCMAG with $d_{min} > 0$ is significantly smaller than the size of clusters found by the baseline methods. For example, the baseline methods find 12 gigantic clusters which contain more than 8753 (50%) movies on IMDB graph, and find 110 gigantic clusters which contain more than 40,000 (50%) authors on bibliographic graph. Such gigantic clusters do not correspond to densely connected communities and are of little significance. (2) The number of clusters found by SCMAG with $d_{min} > 0$ is orders of magnitude smaller than that found by the baseline methods. Many clusters with a sparse connectivity are pruned by our method. This result confirms that the connectivity consideration is necessary for subspace clustering on attributed graphs.

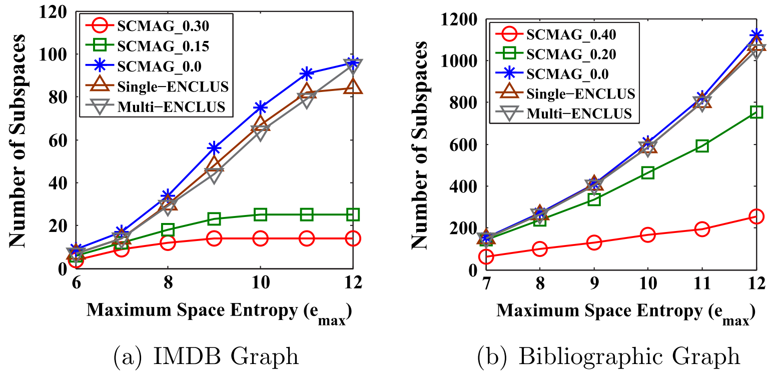


Fig. 10. Number of subspaces found.

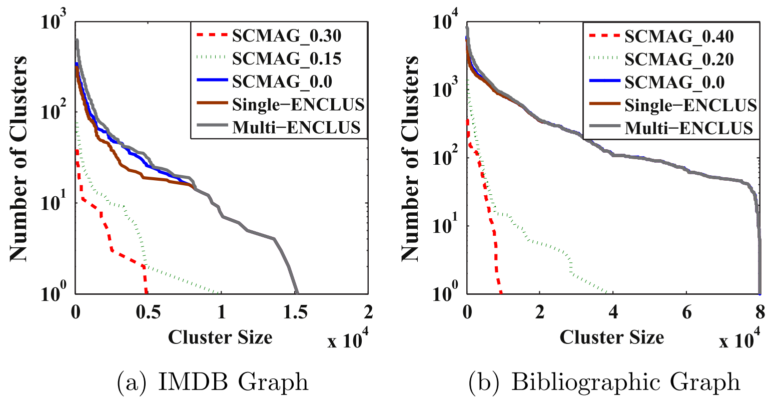


Fig. 11. Clustering size vs. number.

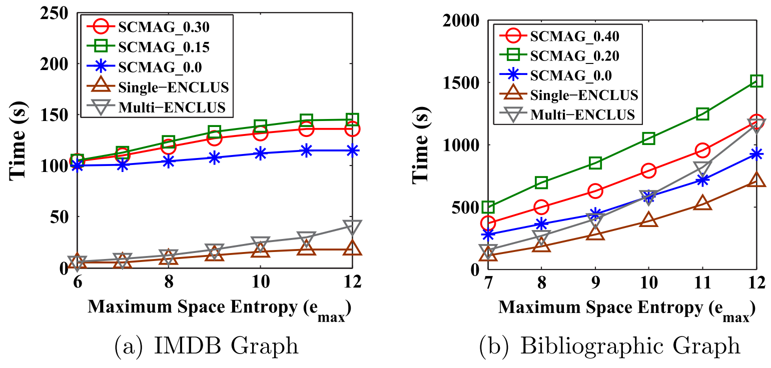


Fig. 12. Clustering efficiency.

5.6. Clustering efficiency comparison

We compare the efficiency of different subspace clustering algorithms by varying the parameter e_{max}. Fig. 12 shows the running time on IMDB and bibliographic graphs. Since SCMAG has an extra cost of random walk computation, it is 12.5 times slower on average than Single-ENCLUS on IMDB and 2.4 times slower on average on bibliographic graph.

5.7. Comparison with SA-cluster

In this experiment, we compare our method with SA-Cluster [6] on IMDB (Table 13) and bibliographic graph (Table 14). Since Inc-Cluster [37] generates the same clustering results as SA-Cluster, we do not test Inc-Cluster. We set $d_{min} = 0.30$ on IMDB and $d_{min} = 0.40$ on bibliographic graph. SA-Cluster considers the full attribute space for clustering. We set the

Table 13
SA-Cluster vs. SCMAG: IMDB.

	SA-Cluster	SCMAG
Cluster number	39	39
Max cluster size	16,685(95%)	4857(28%)
Avg cluster size (except max one)	22	732
Entropy	1.741	0.598
Density	0.205	0.452

Table 14
SA-Cluster vs. SCMAG: bibliographic.

	SA-Cluster	SCMAG
Cluster number	362	362
Max cluster size	73,595(92%)	9494(12%)
Avg cluster size (except max one)	18	1893
Entropy	2.048	1.316
Density	0.357	0.461

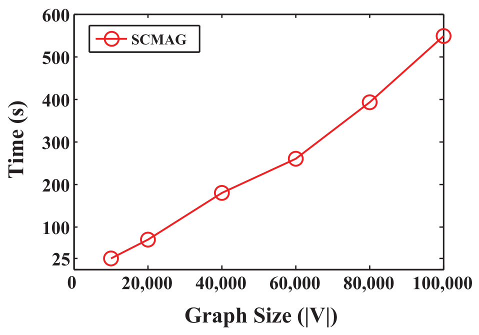


Fig. 13. Scalability evaluation vs. vertex number $|V|$.

parameter of cluster number K in SA-Cluster to be the same as the number of clusters found by SCMAG. From both tables we can see SA-Cluster always outputs a gigantic cluster containing more than 90% nodes, which does not make much sense. The remaining clusters are extremely small, i.e., the average size of clusters except the gigantic one is 22 on IMDB and 18 on bibliographic graph. On the other hand, the clusters found by SCMAG are not extremely huge or small, as each cluster size has to exceed the minimum coverage v_{min}. In terms of cluster quality, SCMAG significantly outperforms SA-Cluster on both entropy and density. These results confirm that subspace clustering on attributed graphs generates high-quality clusters, i.e., with high structural connectivity and high attribute similarity.

5.8. Synthetic data

In this experiments, we use synthetic graph datasets to conduct scalability test and quality evaluation. Each graph has the number of edges as $|E| = 4|V|$, where $|V|$ is the vertex number. A vertex is associated with 15 attributes, of which 10 attributes are categorical and multi-valued. The domain of each attribute is 20.

5.8.1. Scalability test

To evaluate the scalability of SA-Cluster, we generate a series of synthetic graphs described above by increasing the number of vertices $|V|$ from 10,000 to 100,000. We set the parameters $e_{max} = 10, i_{min} = 0.01, v_{min} = 50$ and $d_{min} = 0.3$. We report the running time result in Fig. 13. As we can see, SA-Cluster scales very well with the increasing vertex number. The running time on the graph with $|V| = 100,000$ by SA-Cluster is only 22 times slower than the cost on the graph with $|V| = 10,000$, due to the effective pruning techniques.

5.8.2. Quality evaluation

To evaluate clustering quality by all methods, we generate a series of synthetic graphs with ground-truth clusters. Each graph contains 5 clusters of 10 nodes by default. Each cluster has a density of 0.4, and 5 relevant dimensions out of 15

Table 15
F1-score of clusters produced by different methods on ground-truth synthetic graphs.

Cluster size	GAMer	Single-ENCLUS	Multi-ENCLUS	SCMAG
10	0.97	0.60	0.60	0.98
15	0.86	0.53	0.52	0.98
20	–	0.45	0.45	0.97
25	–	0.41	0.40	0.97
30	–	0.36	0.36	0.98

dimensions. We also add the noise nodes into graphs, which has the same total number of nodes in clusters. Noise nodes do not belong to any clusters. We evaluate the quality of clusters on different graphs by increasing the number of clusters size.The clustering quality is measured by F1-score criterion [13]. The results are reported in Table 15. For all graphs, the parameters of each method are consistly set up. GAMer uses parameters $n_{min} = 8, \gamma_{min} = 0.3$, and other parameters are set to the default values in Table 4. SCMAG uses parameters $e_{max} = 10, i_{min} = 0.01, v_{min} = 8$ and $d_{min} = 0.3$. Single-ENCLUS and Multi-ENCLUS both use the same parameters $e_{max} = 10, i_{min} = 0.01$ and $v_{min} = 8$.

As we can see, our method SCMAG achieved the highest F1-scores among all methods on all graphs. For the graphs with cluster size of 10 and 15, GAMer obtained higher scores than Single-ENCLUS and Multi-ENCLUS, due to the consideration of structural density in clusters by GAMer. For the size of clusters exceeding 20 nodes, the program of GAMer failed to produce results by throwing an exception “java.lang.OutOfMemoryError”, which may be resulted from expensive search space.

6. Conclusions

We studied subspace clustering on multi-valued attributed graph for community detection. Different from existing attributed graph clustering algorithms which consider the full space, we find interesting subspaces with good clustering and discover clusters with dense connectivity, homogeneous attribute values and good coverage. We use random walk with restart on an attribute augmented graph to measure the structural closeness and attribute similarity. An indexing scheme is designed to efficiently calculate cell connectivity. Moreover, we propose a new cell merging strategy and a mechanism to handle multi-valued attributes based on graph neighborhood information.

Experimental results show that SCMAG significantly outperforms existing methods ENCLUS, GAMer and SA-Cluster. Some interesting discovered clusters are presented in case study to show the effectiveness of our method.

Acknowledgments

This work is supported by the Hong Kong Research Grants Council (RGC) General Research Fund (GRF) Project No. CUHK 411211, 418512, 14209314, and the Chinese University of Hong Kong Direct Grant No. 4055015, 4055048.

References

[1] C.C. Aggarwal, C. Procopiuc, J. Wolf, P.S. Yu, J.-S. Park, Fast algorithms for projected clustering, in: SIGMOD, 1999, pp. 61–72.
[2] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, in: SIGMOD, 1998, pp. 94–105.
[3] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, S. Vigna, Four degrees of separation, in: WebSci, 2012, pp. 33–42.
[4] L. Bo, X. Wu, Z.M. Zhang, P.S. Yu, Unsupervised learning on k-partite graphs, in: KDD, 2006, pp. 317–326.
[5] C.H. Cheng, A.W. Fu, Y. Zhang, Entropy-based subspace clustering for mining numerical data, in: KDD, 1999, pp. 84–93.
[6] H. Cheng. Y. Zhou, J.X. Yu, Graph clustering based on structural/attribute similarities, in: VLDB, 2009, pp. 718–729.
[7] D. Coppersmith, S. Winograd, Matrix Multiplication via Arithmetic Progressions, in: J. Symb. Comput., 1990, pp. 251–280.
[8] W. Cui, Y. Xiao, H. Wang, Y. Lu, W. Wang, Online search of overlapping communities, in: SIGMOD, 2013, pp. 277–288.
[9] P. Desikan, N. Pathak, J. Srivastava, V. Kumar, Incremental page rank computation on evolving graphs, in: WWW, 2005, pp. 1094–1095.
[10] I.S. Dhillon, Y. Guan, B. Kulis, Weighted graph cuts without eigenvectors a multilevel approach, in: IEEE Trans. TPAMI, 2007, pp. 1944–1957.
[11] S. Günnemann, B. Boden, T. Seidl, Db-CSC: a density-based approach for subspace clustering in graphs with feature vectors, in: ECML/PKDD (1), 2011, pp. 565–580.
[12] S. Günnemann, I. Färber, B. Boden, T. Seidl, GAMer: a synthesis of subspace clustering and dense subgraph mining, Knowl. Inf. Syst. 40 (2) (2014) 243–278.
[13] S. Günnemann, I. Färber, B. Boden, T. Seidl, Subspace clustering meets dense subgraph mining: a synthesis of two paradigms, in: ICDM, 2010, pp. 845–850.
[14] X. Huang, H. Cheng, L. Qin, W. Tian, J.X. Yu, Querying k-truss community in large and dynamic graphs, in: SIGMOD, 2014, pp. 1311–1322.
[15] A. Leman, H. Tong, B. Meeder, C. Faloutsos, PICS: parameter-free identification of cohesive subgroups in large attributed graphs, in: SDM, 2012, pp. 439–450.
[16] G. Jeh, J. Widom, SimRank: a measure of structural-context similarity, in: KDD, 2002, pp. 538–543.
[17] B. Long, Z.M. Zhang, X. Wu, P.S. Yu, Spectral clustering for multi-type relational data, in: ICML, 2006, pp. 585–592.
[18] B. McWilliams, G. Montana, Subspace clustering of high-dimensional data: a predictive approach, Data Min. Knowl. Discov. 28 (3) (2014) 736–772.
[19] F. Moser, R. Colak, A. Rafiey, M. Ester, Mining cohesive patterns from graphs with feature vectors, in: SDM, 2009, pp. 593–604.
[20] H.S. Nagesh, S. Goil, A.N. Choudhary, A scalable parallel subspace clustering algorithm for massive data sets, in: ICPP, 2000, pp. 477.
[21] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2004) 026113.
[22] L. Parsons, E. Haque, H. Liu, Fast algorithms for projected clustering, in: Workshop on Clustering High Dimensional Data and its Applications, SDM, 2004, pp. 48–56.
[23] D. Pham, D.Q. Phung, B. Saha, S. Venkatesh, Sparse subspace clustering via group sparse coding, in: SDM, 2013, pp. 130–138.

[24] P. Pons, M. Latapy, Computing communities in large networks using random walks, J. Graph Algor. Appl. 10 (2) (2006) 191–218.
[25] C.M. Procopiuc, M. Jones, P.K. Agarwal, T.M. Murali, A monte carlo algorithm for fast projective clustering, in: SIGMOD, 2002, pp. 418–427.
[26] V. Satuluri, S. Parthasarathy, Scalable graph clustering using stochastic flows: Applications to community discovery, in: KDD, 2009, pp. 737–746.
[27] V. Satuluri, S. Parthasarathy, Y. Ruan, Local graph sparsification for scalable clustering, in: SIGMOD, 2011, pp. 721–732.
[28] S.E. Schaeffer, Graph clustering, in: Computer Science Review, 2007, pp. 27–64.
[29] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Machine Intell. 22 (8) (2000) 888–905.
[30] K. Sim, V. Gopalkrishnan, A. Zimek, G. Cong, A survey on enhanced subspace clustering, Data Min. Knowl. Discov. 26 (2) (2013) 332–397.
[31] Y. Tian, R.A. Hankins, J.M. Patel, Efficient aggregation for graph summarization, in: SIGMOD, 2008, pp. 567–580.
[32] H. Tong, C. Faloutsos, B. Gallagher, T. Eliassi-Rad, Fast best-effort pattern matching in large attributed graphs, in: KDD, 2007, pp. 737–746.
[33] H. Tong, C. Faloutsos, J. Pan, Fast random walk with restart and its applications, in: ICDM, 2006, pp. 613–622.
[34] X. Xu, N. Yuruk, Z. Feng, T.A.J. Schweiger, Scan: a structural clustering algorithm for networks, in: KDD, 2007, pp. 824–833.
[35] J. Yang, J.J. McAuley, J. Leskovec, Community detection in networks with node attributes, in: ICDM, 2013, pp. 1151–1156.
[36] A.W. Yu, N. Mamoulis, H. Su, Reverse top-k search using random walk with restart, in: PVLDB 7(5), 2014, pp. 401–412.
[37] Y. Zhou, H. Cheng, J.X. Yu, Clustering large attributed graphs: an efficient incremental approach, in: ICDM, 2010, pp. 689–698.