Clustering Attributed Multi-graphs with Information Ranking

Andreas Papadopoulos, Dimitrios Rafailidis,
George Pallis, and Marios D. Dikaiakos

Department of Computer Science, University of Cyprus
`{andpapad,drafail,gpallis,mdd}@cs.ucy.ac.cy`

Abstract. Attributed multi-graphs are data structures to model real-world networks of objects which have rich properties/attributes and they are connected by multiple types of edges. Clustering attributed multi-graphs has several real-world applications, such as recommendation systems and targeted advertisement. In this paper, we propose an efficient method for Clustering Attributed Multi-graphs with Information Ranking, namely CAMIR. We introduce an iterative algorithm that ranks the different vertex attributes and edge-types according to how well they can separate vertices into clusters. The key idea is to consider the 'agreement' among the attribute- and edge-types, assuming that two vertex properties 'agree' if they produced the same clustering result when used individually. Furthermore, according to the calculated ranks we construct a unified similarity measure, by down-weighting noisy vertex attributes or edge-types that may reduce the clustering accuracy. Finally, to generate the final clusters, we follow a spectral clustering approach, suitable for graph partitioning and detecting arbitrary shaped clusters. In our experiments with synthetic and real-world datasets, we show the superiority of CAMIR over several state-of-the-art clustering methods.

Keywords: attributed multi-graphs, spectral clustering, information ranking

1 Introduction

Information networks in many different domains such as biology, telecommunications, software engineering and social networking, can be modelled as attributed multi-graphs. In an attributed multi-graph, vertices have various attributes and are connected by multiple types of edges, such as user interactions and personal relations. Clustering attributed multi-graph methods aim at grouping related vertices into clusters by optimizing different objective functions, e.g. by minimizing the entropy and/or maximizing the connectivity and the density of the generated clusters. Recent works have shown that taking into account both the network structure and the vertex attributes can lead to improved clustering accuracy [1].

In real-world networks, each vertex property, i.e. attribute and edge-type, contains different information, while some of these properties may be irrelevant to the clustering task. For instance, clustering a bibliography network, authors' attribute 'area of interest' is important, while attributes 'name' and 'gender' may introduce noise and thus reduce the clustering accuracy. Hence, the significance of each attribute and edge-type must be identified. One way to achieve this is to rank the vertex attributes and the edge-types, providing thus for each vertex property different importance in the clustering [1, 2].

Recently, many methods for clustering attributed graphs have been proposed [1], [3], [4], [5]. These works use unified distance/similarity measures or models that combine the vertex structural and attribute properties. They aim to partition attributed graphs into dense clusters with vertices having similar attributes, where attribute similarity is modelled by low entropy within each generated cluster. However, using weights to capture the different importance of vertex attributes in attributed graphs has received limited attention [1], [6], while several methods ignore the existence of multiple edge-types. HASCOP [2] is one approach that considers the different importance of vertex attributes and multiple edge-types. Nonetheless, similarly to the other methods, the multiple edge-types and the vertex attributes are considered separately to perform the weighting. Therefore, a mechanism to identify the importance of each vertex attribute and edge-type for attributed multi-graphs is required.

In this paper, we consider the aforementioned challenges and we propose *a new approach for Clustering Attributed Multi-graphs with Information Ranking*, namely CAMIR. Specifically, we present *a mechanism to rank and consequently weigh the vertex properties in attributed multi-graphs*, by iteratively co-regularizing the clustering hypotheses across the vertex properties. Co-regularization is a well-known technique [7], which we use to compute the 'agreement' among the vertex attributes and the edge-types in attributed multi-graphs. Two vertex properties 'agree' if they assign vertices the same cluster labels when they are used individually. The vertex property with the highest agreement best separates the vertices into clusters, while the property with the lowest agreement introduces noise and reduces the clustering accuracy. In our method, we rank all vertex properties accordingly; that is, we assign the highest and the lowest ranks to the properties with the highest and the lowest agreements respectively. Taking advantage of our ranking mechanism, we *assign a weight parameter to each vertex property to compute a unified similarity measure for attributed multi-graphs*. Finally, we follow *a spectral clustering approach to partition the attributed multi-graph* and to generate the final clusters. The reason for using spectral clustering is that it identifies clusters of arbitrary shapes and sizes, by performing clustering in the eigenspace. Our experimental evaluation on synthetic and real-world data sets, i.e. two bibliography networks and a network of software packages available on Google code repository, shows that the proposed *CAMIR method outperforms several state-of-the-art clustering algorithms.*

The remainder of this paper is organized as follows, Section 2 describes the related work, and in Section 3 the problem of clustering attributed multi-graphs is formally defined. Section 4 presents the proposed CAMIR method and Section 5 demonstrates our experimental evaluation. Finally, Section 6 concludes the paper.

2 Related Work

Several clustering methods have been proposed for plain graphs (with a single edge-type) and multi-graphs [8–10]. However, these methods ignore the case of having vertex attributes and cannot be directly applied to attributed graphs. Representative approaches of attributed graph clustering methods are divided into two main categories: (a) distance-based [1], [6] and (b) model-based [5], [11]. Also, methods for clustering attributed graphs identify clusters either in the full space of the network [5], [6], [12] or in multiple sub-spaces [4], [13], [14]. In this paper, we focus on the case of clustering attributed graphs in the full space.

2.1 Distance-based Clustering

SACluster [6] is a distance-based clustering method for attributed graphs, which computes a unified distance measure to capture both the structural and attribute similarities of the vertices. The key idea is to build an attribute augmented graph, equal to the initial graph enriched with new vertices each of which represents an attribute value. An edge from a graph vertex to an attribute vertex is added on the condition that the graph and attribute vertices have the same attribute value. The weight of the new edge depends on the importance that each attribute has. SACluster uses random walk on the augmented graph to measure the distance between two vertices, computing thus a unified distance measure. Several extensions of SACluster have been proposed to minimize the computational cost of random walks, such as SA-Cluster-Opt [15] and Inc-Cluster [1]. However, SACluster and its extensions do not work in multi-graphs, but only in attributed graphs.

PICS [3] identifies clusters with high similar connectivity, i.e., a cluster where the neighborhoods of its vertices highly overlap. The goal of PICS is to compress the adjacency, formed by the edges, and the attribute matrices that represent the attributed graph, by minimizing the cost of encoding in bits the clustering result. To minimize the encoding cost, PICS rearranges the rows and the columns of the adjacency and attribute matrices, so that all rows representing vertices in the first cluster are listed first, followed by rows representing vertices in the second cluster, and so on. At each iteration, PICS splits the cluster with the maximum entropy per attribute/edge, until a new splitting increases the total encoding cost. PICS does not consider multiple edge-types and consequently does not work in multi-graphs.

HASCOP [2] is an iterative algorithm for clustering attributed multi-graphs. HASCOP computes a heuristic unified distance function that combines the structural and attribute properties of the vertices based on their importance. The unified similarity of the vertices is defined as the product of their structural and attribute similarities. It tries to optimize similar connectivity and attribute homogeneity. Initially, each vertex is assigned to a singleton cluster. At each iteration, vertices with the highest unified similarities form the new clusters. The algorithm terminates if the number of clusters is not reduced further at the end of an iteration. Also, HASCOP considers the different importance of the attributes and edge-types over the iterations. However, HASCOP has high computational cost for re-evaluating weights at each iteration.

Spectral clustering of attributed graphs constructs a graph, where the edges between the vertices represent the vertex similarities based on their attributes and connections. Then, spectral clustering solves a relaxation of the normalized min-cut problem on the constructed graph to generate the final clusters [7]. Also, spectral clustering on attributed graphs has been used to identify clusters in the subspace projections of the attribute data; e.g. vertices that are close based on a subset of their attributes are grouped together [13, 14].

2.2 Model-based Clustering

Model-based methods enforce the intra-cluster similarity by modelling the attribute values and the connections/edges of vertices in a cluster by various distributions. For instance, BAGC [12] uses Bayesian inference, considering that the vertices in the same cluster should follow a common multinomial distribution for each of their attributes and a common Bernoulli distribution for their connections. BAGC starts with a random assignment of the vertices into clusters. Then, the parameters of all the distributions are iteratively recalculated. Vertices are assigned to a cluster, if vertex attributes and connections follow the respective attributes and connections distributions of the cluster. GBAGC [5] extends BAGC so as to handle weighted attributed graphs. The big advantage of BAGC and GBAGC is their scalability; however, they do not work for multi-graphs. CESNA [11] defines a model on attributed graphs that also enforces the intra-cluster similarities. CESNA models vertex attributes and connections in the same cluster with Bernoulli distributions. Nonetheless, CESNA differs from BAGC and all the aforementioned methods, by identifying overlapping communities.

3 Problem Definition

Our notation is presented in Table 1. Following standard notations, sets are denoted by calligraphic upper case letters, e.g., $\mathcal{A}$; matrices by plain upper case letters, e.g., A; functions by lower case calligraphic letters, e.g. $\mathscr{a}$; numbers and set elements by lower case letters, e.g. a.

An attributed multi-graph G is a set of $|\mathcal{T}|$ different attributed graphs defined over the same set of vertices, where $\mathcal{T} = \{t_i\}$ is the set of edge-types. Formally, $G = \{G_t(\mathcal{V}, \mathcal{E}_t)\}_{t=1}^{\mathcal{T}}$, where $\mathcal{V} = \{v_i\}$ is the set of vertices; $\mathcal{E}_t = \{(v_i, v_j) : v_i, v_j \in \mathcal{V}\}$ is the set of all edges of type t; $\omega_t : \mathcal{E}_t \rightarrow (0, 1]$ returns the weight of the edges of type t, i.e $\omega_t(v_i, v_j)$ is the weight of the edge of type t from v_i to v_j. Each vertex has $|\mathcal{A}|$ different attribute values, where $\mathcal{A} = \{\alpha_i : 1 \leq i \leq |\mathcal{A}|\}$

Table 1: Notations

Symbol	Description
$\mathcal{V}$	Set of vertices $V = \{v_i : 1 \leq i \leq \|\mathcal{V}\|\}$
$\mathcal{E}_t$	Set of edges of type t. $\mathcal{E}_t = \{(v_i, v_j) : v_i, v_j \in \mathcal{V}\}$
$\omega_t : \mathcal{E}_t \rightarrow (0, 1]$	Function that returns the weight of the edges of type t
$\mathcal{A}$	Set of attributes $A = \{\alpha_i : 1 \leq i \leq \|\mathcal{A}\|\}$
$a_\alpha(v_i)$	Value of vertex v_i on attribute α
$\mathcal{T}$	Set of edge-types, with $\mathcal{T} = \{t_i : 1 \leq i \leq \ \|\mathcal{T}\|, \mathcal{E}_t \neq \emptyset\}$
$\mathcal{P} = \{\mathcal{A} \cup \mathcal{T}\}$	Set of all vertex properties (attributes and edge-types)
$w : \mathcal{P} \rightarrow (0, 1]$	Function that returns the importance/weight of a vertex property p
k	Number of clusters
$S^p \in \mathbb{R}^{\|\mathcal{V}\| \times \|\mathcal{V}\|}$	Similarity matrix for property p
$L^p \in \mathbb{R}^{\|\mathcal{V}\| \times \|\mathcal{V}\|}$	Normalized Laplacian matrix of S^p
$U^p \in \mathbb{R}^{\|\mathcal{V}\| \times k}$	First k eigenvectors of L^p

is the set of all vertex attributes. The attribute values of a vertex are given by the functions a_α, i.e. $a_\alpha(v_i)$ is the attribute value of vertex v_i on attribute α. Also, we denote as $\mathcal{P} = \{p : p \in \mathcal{T} \vee p \in \mathcal{A}\}$ the set of all vertex properties, i.e. edge-types and attributes.

The problem of clustering attributed multi-graphs is defined as follows: given an attributed multi-graph G with $|\mathcal{P}|$ different vertex properties and a number of clusters k, the goals are (a) to compute a weight $w(p)$ for each vertex property $p \in \mathcal{P}$, in order to construct a unified similarity matrix $S \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$; (b) to generate the k clusters based on S, by maximizing the similarity of vertices within a cluster and minimizing the similarity between vertices in different clusters.

4 The Proposed CAMIR Method

4.1 Overview

The proposed CAMIR method consists of the following steps: (a) the attributed multi-graph is processed to rank the vertex properties and to calculate their weights accordingly; then a unified similarity measure is computed by considering all vertex properties and their importance based on the calculated weights. (b) a spectral clustering approach is adopted to embed the vertices into the respective eigenspace of the unified similarity measure. The reason for selecting spectral clustering is that it identifies clusters of arbitrary shapes and sizes. The key idea in spectral clustering is to achieve graph partitioning by finding the best cut. Spectral clustering finds the best cut by performing eigendecomposition of a graph Laplacian matrix, constructing thus a k-dimensional eigenspace. Spectral clustering methods differ in how they define and construct the Laplacian matrix L of the similarity matrix S and, thus, which eigenvectors are selected to represent the graph, aiming to exploit special properties of different matrix formulations [16]. For the interested reader, Ulrike von Luxburg's tutorial [16] includes examples of different Laplacians' constructions. Moreover, different objective functions are used to derive the best cut. For example, Ratio Cut [17] tries to minimize the total cost of the edges crossing the cluster boundaries, normalized by the size of the k clusters, to encourage balanced cluster sizes. Normalized Cut (NCut) [18] uses the same objective criterion as Ratio Cut, normalized by the total degree of each cluster, making thus the clusters having similar degrees. In our approach we followed the NCut method.[1] In the following Sections we present each step of the proposed CAMIR method in detail.

4.2 Information Ranking

Given an attributed multi-graph G, we compute the affinity matrices $S^p \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where $S^p_{ij} \geq 0$ denotes the relationship - similarity between v_i and v_j for property $p \in \mathcal{P} \equiv \{\mathcal{A} \cup \mathcal{T}\}$. For each edge-type $t \in \mathcal{T}$, the respective similarity matrix is calculated as follows:

$$S^t_{ij} = \omega_t(v_i, v_j) \tag{1}$$

For each vertex attribute $a \in \mathcal{A}$ we calculate the respective similarity matrix based on the Gaussian kernel:[2]

$$S^\alpha_{ij} = \exp\left(-\frac{||a_\alpha(v_i) - a_\alpha(v_j)||^2}{2 \cdot \sigma_i \cdot \sigma_j}\right) \tag{2}$$

where σ_i is a scaling parameter to control how rapidly the similarity S^α_{ij} reduces according to the distance between v_i and v_j. For each vertex v_i, the scaling parameter σ_i allows the self-tuning of the vertex-to-vertex distances according to the local statistics of the neighbourhoods surrounding v_i. We followed the self-tuning strategy of [19]. Provided that v_i has ϵ neighbours, σ_i is calculated as the average of the ϵ distances.

According to Equations (1) and (2), for each vertex property $p \in \mathcal{P}$ we construct an affinity matrix, denoted as $S^p \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. We compute the normalized Laplacian $L^p \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ of S^p as follows:

$$L^p = I - D^{p^{-1/2}} \cdot S^p \cdot D^{p^{-1/2}} \tag{3}$$

where I is the identity matrix and D^p is a diagonal matrix calculated as follows:

$$D^p_{ii} = \sum_{j=1}^{|\mathcal{V}|} S^p_{ij} \tag{4}$$

[1] Alternatively, several parallel spectral clustering methods could be used in the proposed approach, such as the works of [19], [20], to reduce the computational time of spectral clustering.

[2] Also, other types of kernel functions could be used, such as linear and polynomial, thoroughly examined in [21] for machine learning methods.

where $D^{p^{-1/2}}$ indicates the inverse square root of D^p. For any S with $S_{ij} \geq 0$, the Laplacian matrix is symmetric positive semi-definite [16]. After computing the $|\mathcal{P}|$ different normalized Laplacians, we perform eigendecomposition on each of the normalized Laplacians to retrieve their top k eigenvectors, denoted by $U^p \in \mathbb{R}^{|\mathcal{V}| \times k}$ for the p-th Laplacian L^p.

According to [7] the vertex property p that best separates the vertices is selected using the following equation:

$$f(\mathcal{P}) = \underset{U^p \in \mathbb{R}^{|\mathcal{V}| \times k}}{\arg\max}\ tr \left[U^{p^T} \cdot \left(L^p + \lambda \cdot \sum_{\substack{i=1 \\ p_i \neq p}}^{|\mathcal{P}|} \left(U^{i^T} \cdot U^i \right) \right) \cdot U^p \right] \tag{5}$$

where $\mathcal{P}$ is the set of all vertex properties, tr is the trace of the matrix, U^{i^T} is the transpose matrix of U^i, and λ is a co-regularization parameter that controls the penalization of a property according to its 'disagreement' with the other properties,[3] denoted by the sum in Equation (5). According to [7] two different vertex properties 'agree' if they produced the same clustering result when used individually. Equation (5) returns the property p that has the highest 'agreement' with the rest properties, assuming that if we use only the selected property p for clustering we expect to find accurate clusters, independently of the rest of the properties. The mathematical proof can be found in [7].

However, considering just a single property to perform clustering contradicts with works elaborating on the use of all vertex properties to improve clustering accuracy [1], [4]. We propose to iteratively apply Equation (5) $|\mathcal{P}|$ times. At each iteration we exclude the properties that have been already ranked. In particular, given the set of unranked properties, denoted as $\mathcal{P}_u$, we compute the ranking $r(p)$ of a property p by the following equation:

$$r(p) = \begin{cases} |\mathcal{P}_u| : |\mathcal{P}_u| > 1 \wedge f(\mathcal{P}_u) = U^p \\ 1 \quad\;\; : |\mathcal{P}_u| = 1 \end{cases} \tag{6}$$

In doing so, the highest rank is given to the property with the highest 'agreement' with the other properties; while the lowest rank is given to the property that is selected last, does not 'agree' with the rest of properties, and consequently introduces noise to the clustering. Equation (6) maps the vertex properties to the ranks $\{|\mathcal{P}|, |\mathcal{P}| - 1, \ldots, 1\}$ according to the order the properties are selected. To calculate the weight of each property, we perform a normalization of the properties' ranking as follows:

$$w(p_i) = \frac{r(p_i)}{\sum_{j=1}^{|\mathcal{P}|} r(p_j)} \tag{7}$$

[3] Following [7] we use a common λ for all properties. In practice though, $\lambda = 0.001$ is an appropriate value to control the impact of the other properties, as we observed in our experiments.

Algorithm 1 CAMIR Algorithm

```
Input: Attributed multi-graph $G$, number of clusters $k$
Output: Cluster labels
  $\mathcal{P}_u = \mathcal{P}$
  while $|P_u| \neq \emptyset$ do
    Rank a property $p$ using Equation (6)
    $\mathcal{P}_u = \mathcal{P}_u \setminus \{p\}$
  end while
  Compute weights $w(p_i)$ based on Equation (7)
  Compute the unified similarity matrix $S$ using Equation (8)
  Compute the normalized Laplacian $L$ of $S$ according to Equation (3)
  Perform eigendecomposition on $L$ to obtain $U$ - the top $k$ eigenvectors
  Run $k$-means on $U$ to generate the final cluster labels
```

Equation (7) assigns higher weights to more important vertex properties and down-weighs the properties that have lower ranks and introduce noise over the clustering. Finally, according to property weights we compute the unified similarity matrix $S \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ as the weighted sum of the $|\mathcal{P}|$ similarity matrices S^p:

$$S = \sum_{\forall p \in \mathcal{P}} w(p) \cdot S^p \tag{8}$$

4.3 Generating the Final Clusters

In our method, spectral clustering is formulated as follows: given the $|\mathcal{V}|$ vertices of the attributed multi-graph G and the unified similarity matrix $S \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, calculated based on Equation (8), the goal is to find k disjoint vertex subsets, namely clusters, whose union is the whole data set, by solving the following standard eigendecomposition problem:

$$\underset{U \in \mathbb{R}^{|\mathcal{V}|\times k}}{\arg\max}\, tr\left(U^T \cdot L \cdot U\right),\ \text{s.t. } U^T \cdot U = I \tag{9}$$

where L is the normalized Laplacian of the unified similarity matrix S given by Equation (8). The columns of $U \in \mathbb{R}^{|\mathcal{V}|\times k}$ are the top k eigenvectors of the Laplacian matrix L, while its rows are the embeddings of the $|\mathcal{V}|$ vertices in the k-th dimensional eigenspace. The final k clusters are generated by applying k-means algorithm to the $|\mathcal{V}|$ embeddings of the vertices. Algorithm 1 presents the pseudocode of the two-step CAMIR algorithm.

5 Experiments

5.1 Datasets

In our experiments we use synthetic and three real-world datasets, summarized in Table 2.

Table 2: Real-world datasets

Dataset	$\|\mathcal{V}\|$	$\|E\|$	$\|\mathcal{A}\|$	$\|\mathcal{T}\|$	$\|\mathcal{P}\|$
DBLP-1K	1000	17128	2	1	3
DBLP-10K	10000	65734	2	1	3
GoogleSP-23	1297	268956	5	2	7

Synthetic datasets have been generated using a modified version of the generator in [4]. The $|\mathcal{V}|$ vertices are divided into k groups. F or each group two parameters specify its similar connectivity and attribute homogeneity. If a vertex in a group connects to a vertex u, then Similar Connectivity parameter specifies the least fraction of vertices in the group that also connect to vertex u. Attribute Homogeneity parameter specifies the least fraction of vertices in a group sharing the same attribute value. The values of the attributes are drawn from a Bernoulli distribution. We set Similar Connectivity and Attribute Homogeneity parameters to 0.8 and, unless stated otherwise, the synthetic datasets have: 1000 vertices, 25 clusters, 1 edge-type and each vertex is characterized by 4 attributes. The reason for selecting 1 edge-type is that many competitors, e.g. SACluster [6], BAGC [12], and PICS [3], do not work on multi-graphs, but only on attributed graphs (Section 2).

DBLP-1K and DBLP-10K datasets[4] consist of 1000 and 10000 vertices, respectively. The vertices represent the top authors from the complete DBLP dataset. An author has the following properties: publications and the primary area of interest, e.g. databases, data mining, information retrieval, artificial intelligence, etc. A weighted edge between two authors represents the number of publications they have co-authored. In both datasets the clusters are unknown.

GoogleSP-23 is a dataset constructed by crawling the file system of a virtual machine in which 23 software packages were downloaded from the Google code repository[5]. This dataset was also used in the work of [2]. A vertex represents a software file and has the following attributes: file size, last access time, modification time, creation time and file-type. There are two types of edges in this dataset based on the file name and file path similarities. Given two files we calculate the distance of their file system paths using a string edit distance algorithm. An edge between the two files is added if their paths' distance is less than the average paths distance in the network. Similarly, the edges based on names are added. Thus, GoogleSP-23 is an attributed multi-graph, where each vertex has 5 attributes and there are 2 edge-types. In this dataset the clusters are known, where each cluster corresponds to a software package.

[4] The full DBLP dataset is available online at `http://kdl.cs.umass.edu/data/dblp/dblp-info.html`.

[5] Available online at `http://code.google.com`

5.2 Evaluation Protocol

In our experiments we use the (a) Normalized Mutual Information (NMI) and (b) entropy metrics. NMI is in the range of $[0, 1]$. High NMI is equivalent to high similarity between the resulted clustering of an examined method and the ground-truth. Given the clustering of an examined method, denoted by $\mathcal{S}_1 = \{\mathcal{B}_1, \ldots, \mathcal{B}_{k1}\}$ and the ground-truth in $\mathcal{S}_2 = \{\mathcal{C}_1, \ldots, \mathcal{C}_{k2}\}$, where it may hold $k1 \neq k2$, NMI is calculated as follows:

$$NMI(\mathcal{S}_1, \mathcal{S}_2) = \frac{H(\mathcal{S}_1) - H(\mathcal{S}_1|\mathcal{S}_2)}{\min(H(\mathcal{S}_1), H(\mathcal{S}_2))} \tag{10}$$

where:

$$H(\mathcal{S}) = -\sum_{i=1}^{|\mathcal{S}|} \frac{|\mathcal{C}_i|}{|\mathcal{V}|} \cdot \log\left(\frac{|\mathcal{C}_i|}{|\mathcal{V}|}\right)$$

$$H(\mathcal{S}_1|\mathcal{S}_2) = -\sum_{i=1}^{|\mathcal{S}_1|}\sum_{j=1}^{|\mathcal{S}_2|} \frac{m_{i,j}}{|\mathcal{V}|} \cdot \log\left(\frac{m_{i,j}/|\mathcal{V}|}{|\mathcal{C}_j|/|\mathcal{V}|}\right)$$

where $m_{i,j}$ is the number of common vertices between clusters $\mathcal{B}_i$ and $\mathcal{C}_j$. In case $NMI(\mathcal{S}_1, \mathcal{S}_2) = 1$, then the two clusterings are identical. At this point we must mention that NMI can be calculated, only if the ground-truth is available. Thus, in the synthetic datasets and GoogleSP-23 we report NMI for each clustering method, while in DBLP-1K and DBLP-10K NMI is not available, since the clusters are unknown.

Entropy ranges in $[0, \infty)$ and is measured for each attribute a as follows:

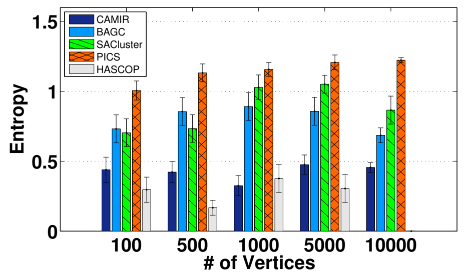
$$entropy(\alpha) = \sum_{j=1}^{k} \frac{|\mathcal{C}_j|}{|\mathcal{V}|} \cdot entropy(\alpha, \mathcal{C}_j) \tag{11}$$

Low entropy is equivalent to high homogeneity between the attributes of the vertices in the same cluster. The overall entropy for a clustering method is the average entropy for all attributes. Thus, the goal of each clustering method is to achieve low overall entropy.

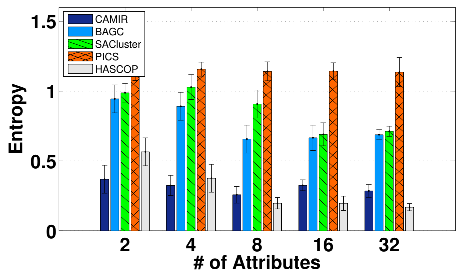
We evaluate the proposed CAMIR method against HASCOP [2], SA-Cluster [6], BAGC [12], and PICS [3]. All experiments were conducted on a desktop pc equipped with a quad core Intel i7 2.8GHz processor and 8GB RAM.

5.3 Evaluation on Synthetic Datasets

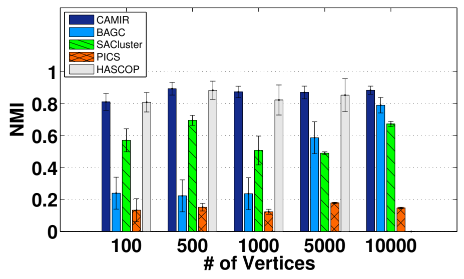
In this set of experiments, we vary the number of vertices in {100, 500, 1000, 5000, 10000} with 4 attributes. Also, we vary the number of attributes in {2, 4, 8, 16, 32} with 1000 vertices. For each variation, we generate 5 random graphs and in Figures 1(a)-(f) we report the average entropy, NMI and computational time out of the 5 runs. We must mention that for number of vertices equal to 10000, we omit the results of HASCOP, since it does not scale due to its heavy



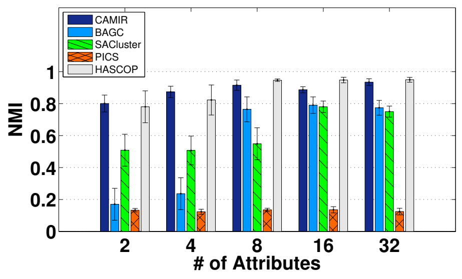
(a) Entropy vs # of vertices



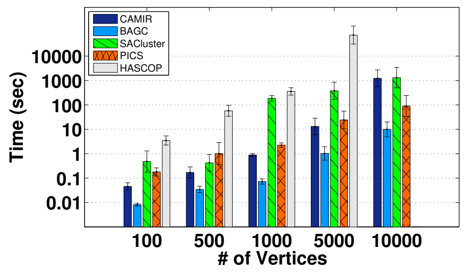
(b) Entropy vs # of attributes



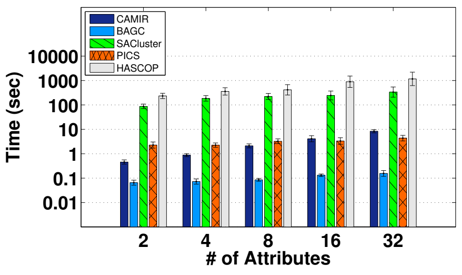
(c) NMI vs # of vertices



(d) NMI vs # of attributes



(e) Runtime vs # of vertices



(f) Runtime vs # of attributes

Fig. 1: Clustering performance on synthetic attributed graphs

computational cost. According to Figures 1(a)-(d) CAMIR outperforms all its competitors, except HASCOP, in terms of entropy and NMI. The high clustering accuracy of CAMIR is based not only on the spectral clustering technique which identifies arbitrary shaped clusters, but also on the proposed weighting mechanism that correctly identifies the importance of the different vertex properties. PICS has high entropy and low NMI, because it converges too early and returns few clusters, by using a self-tuning strategy to determine the number of clusters. HASCOP and CAMIR achieve comparable clustering accuracy, since HASCOP also weighs the vertex properties efficiently. However, HASCOP is much slower than CAMIR, at least two orders of magnitude, as presented

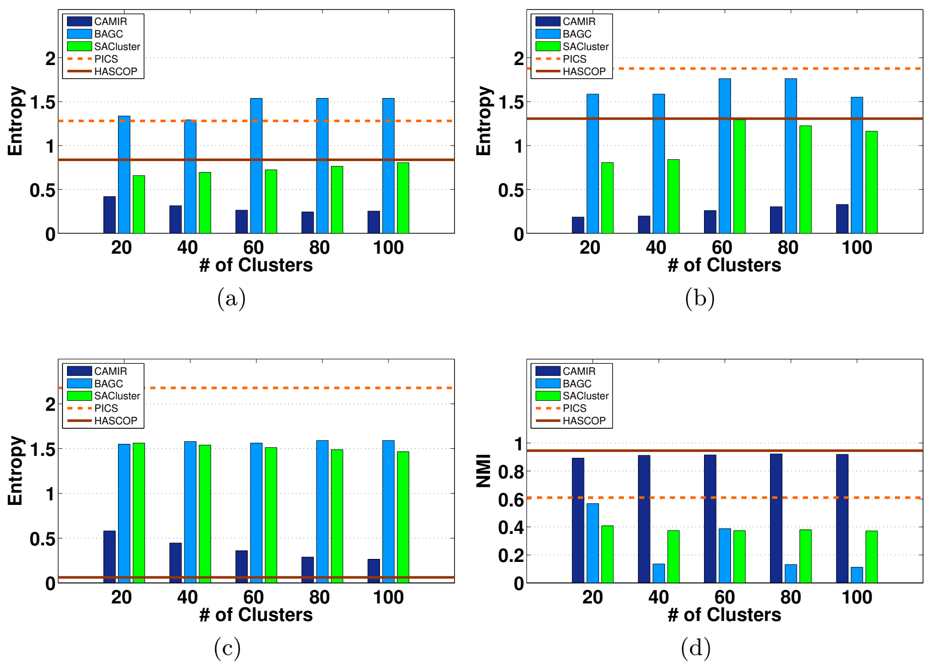


Fig. 2: Performance evaluation on real-world datasets; (a) DBLP-1K (b) DBLP-10K (c)-(d) GoogleSP-23. Since HASCOP and PICS use a self-tuning strategy to determine the number of clusters, both methods are denoted by straight lines

in Figures 1(e)-(f). This happens because HASCOP has a high computational cost of re-evaluating weights at each iteration (Section 2). BAGC is the fastest approach; nevertheless, it has limited clustering accuracy as shown, in Figures 1(a)-(d), since it does not use a weighting scheme to identify the importance of the vertex properties.

5.4 Evaluation on Real-World Datasets

DBLP-1K and DBLP-10K In this set of experiments we vary the number of clusters from $k = 20$ to 100 by a step of 20. PICS and HASCOP use a self-tuning method to determine the number of clusters, returning 8 and 85 clusters in DBLP-1K and 18 and 763 clusters in DBLP10-K, respectively. In Figures 2(a) and (b) we present the average entropy on both datasets. NMI is not available, since the clusters in both datasets are unknown. We observe that in both DBLP datasets CAMIR achieves the best results in terms of entropy for all number of clusters variations. The reason is that CAMIR is robust to noise, by incorporating the importance of the vertex properties according the proposed weighting scheme. Besides the high clustering accuracy, CAMIR requires comparable time to SACluster and PICS, as presented in Table 3. BAGC is the fastest approach

Table 3: Computation time (sec) in the 3 real-world datasets

Algorithm	# of Clusters	DBLP-1K	DBLP-10K	GoogleSP-23
CAMIR	20	0.54	399.13	3.26
	40	0.80	425.15	4.78
	60	1.17	541.39	6.04
	80	1.48	607.81	7.25
	100	2.03	628.94	8.58
BAGC	20	0.12	0.33	0.41
	40	0.13	0.34	0.54
	60	0.14	0.38	0.71
	80	0.14	0.46	0.86
	100	0.16	0.49	0.97
SACluster	20	2.59	313.19	28.84
	40	3.28	407.84	30.11
	60	3.32	466.76	30.57
	80	3.41	474.93	30.81
	100	3.51	503.42	31.86
PICS	8	4.87	-	-
	18	-	495.17	-
	16	-	-	476.49
HASCOP	85	882.18	-	-
	763	-	32957.10	-
	51	-	-	4675.27

but has limited clustering accuracy, denoted by high entropy, while HASCOP is the slowest clustering method.

GoogleSP-23 In this set of experiments, HASCOP and PICS identify 16 and 51 clusters, respectively. Since GoogleSP-23 is an attributed multi-graph we run PICS, SACluster and BAGC separately with only one edge-type and all the vertex attributes, since these methods do not work in multi-graphs, but only in attributed graphs. In Figures 2(c) and (d) we report the best results for the three aforementioned methods. CAMIR handles efficiently the noise in the dataset, by identifying the importance of the vertex properties, resulting thus to low entropy and high NMI. In this set of experiments, HASCOP slightly outperforms CAMIR in terms of entropy; however HASCOP requires a significantly higher computational time than CAMIR, almost 3 orders of magnitude, as presented in Table 3.

6 Conclusion

In this paper we proposed CAMIR, a method for clustering attributed multi-graphs. CAMIR ranks vertex properties of an attributed multi-graph by exploiting the information from edge-types and attributes. Based on the identified ranking CAMIR weighs the vertex properties to construct a unified similarity measure and performs spectral clustering to generate the final clusters. Our experimental evaluation on synthetic and real-world graphs against state-of-the-art attributed graph clustering techniques showed the effectiveness of the proposed CAMIR method, in terms clustering accuracy and computational time.

Real-world information networks are continuously updated. Recently, several incremental spectral clustering strategies have been proposed in the literature [22]. These strategies are able to handle not only insertions and deletions of new objects but also similarity changes between them, by efficiently updating the eigenspace. In our future research we plan to examine both the incremental and evolving strategies of the proposed method in the context of spectral clustering in Big Data [23].

Acknowledgments

This work was partially supported by the EU Commission in terms of the PaaSport 605193 FP7 project (FP7-SME-2013).

References

1. Cheng, H., Zhou, Y., Yu, J.X.: Clustering large attributed graphs: A balance between structural and attribute similarities. ACM Trans. Knowl. Discov. Data **5**(2) (February 2011) 12:1–12:33
2. Papadopoulos, A., Pallis, G., Dikaiakos, M.D.: Identifying clusters with attribute homogeneity and similar connectivity in information networks. In: Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 01. WI-IAT '13, Washington, DC, USA, IEEE Computer Society (2013) 343–350
3. Akoglu, L., Tong, H., Meeder, B., Faloutsos, C.: Pics: Parameter-free identification of cohesive subgroups in large attributed graphs. In: Proceedings of the 12th SIAM International Conference on Data Mining, SDM 2012, SIAM / Omnipress (2012) 439–450
4. Perozzi, B., Akoglu, L., Iglesias Sánchez, P., Müller, E.: Focused clustering and outlier detection in large attributed graphs. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '14, New York, NY, USA, ACM (2014) 1346–1355
5. Xu, Zhiqiang and Ke, Yiping and Wang, Yi and Cheng, Hong and Cheng, James: GBAGC: A general bayesian framework for attributed graph clustering. ACM Trans. Knowl. Discov. Data **9**(1) (2014) 5:1–5:43
6. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. Proc. VLDB Endow. **2**(1) (2009) 718–729
7. Kumar, A., Rai, P., Daume, H.: Co-regularized multi-view spectral clustering. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K., eds.: Advances in Neural Information Processing Systems 24. Curran Associates, Inc. (2011) 1413–1421
8. Karypis, G., Kumar, V.: Multilevel algorithms for multi-constraint graph partitioning. In: Proceedings of the 1998 ACM/IEEE Conference on Supercomputing. SC '98, Washington, DC, USA, IEEE Computer Society (1998) 1–13

9. Papalexakis, E., Akoglu, L., Ience, D.: Do more views of a graph help? community detection and clustering in multi-graphs. In: Information Fusion (FUSION), 2013 16th International Conference on. (2013) 899–905
10. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.J.: SCAN: a structural clustering algorithm for networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '07, New York, NY, USA, ACM (2007) 824–833
11. Yang, J., McAuley, J.J., Leskovec, J.: Community detection in networks with node attributes. [24] 1151–1156
12. Xu, Z., Ke, Y., Wang, Y., Cheng, H., Cheng, J.: A model-based approach to attributed graph clustering. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. SIGMOD '12, New York, NY, USA, ACM (2012) 505–516
13. Günnemann, S., Färber, I., Raubach, S., Seidl, T.: Spectral subspace clustering for graphs with feature vectors. [24] 231–240
14. Gunnemann, S., Boden, B., Farber, I., Seidl, T.: Efficient mining of combined subspace and subgraph clusters in graphs with feature vectors. In: Advances in Knowledge Discovery and Data Mining. Volume 7818 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 261–275
15. Zhou, Y., Cheng, H., Yu, J.X.: Clustering large attributed graphs: An efficient incremental approach. In Webb, G.I., Liu, B., Zhang, C., Gunopulos, D., Wu, X., eds.: ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010, IEEE Computer Society (2010) 689–698
16. Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing **17**(4) (2007) 395–416
17. Chan, P.K., Schlag, M.D.F., Zien, J.Y.: Spectral K-way ratio-cut partitioning and clustering. IEEE Trans. on CAD of Integrated Circuits and Systems **13**(9) (1994) 1088–1096
18. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8) (August 2000) 888–905
19. Chen, W.Y., Song, Y., Bai, H., Lin, C.J., Chang, E.: Parallel spectral clustering in distributed systems. Pattern Analysis and Machine Intelligence, IEEE Transactions on **33**(3) (2011) 568–586
20. Kang, U., Meeder, B., Papalexakis, E.E., Faloutsos, C.: Heigen: Spectral analysis for billion-scale graphs. IEEE Trans. Knowl. Data Eng. **26**(2) (2014) 350–362
21. Hofmann, T., Schölkopf, B., Smola, A.J.: Kernel methods in machine learning. Ann. Statist. **36**(3) (2008) 1171–1220
22. Ning, H., Xu, W., Chi, Y., Gong, Y., Huang, T.S.: Incremental spectral clustering by efficiently updating the eigen-system. Pattern Recogn. **43**(1) (2010) 113–127
23. Mall, R., Langone, R., Suykens, J.A.K.: Kernel spectral clustering for big data networks. Entropy **15**(5) (2013) 1567–1586
24. Xiong, H., Karypis, G., Thuraisingham, B.M., Cook, D.J., Wu, X., eds.: 2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013, IEEE Computer Society (2013)