# Identifying Clusters with Attribute Homogeneity and Similar Connectivity in Information Networks

Andreas Papadopoulos, George Pallis, Marios D. Dikaiakos
Department of Computer Science, University of Cyprus
Email: { andpapad, gpallis, mdd }@cs.ucy.ac.cy

***Abstract*—With the rapid emergence of the internet world, a lot of information networks become available every day. In many cases, these information networks contain objects connected by multiple links and described by different attributes. In this paper the problem of clustering homogeneous information networks in groups with similar attributes and connections is studied. Clustering such networks is a challenging task due to different importance of links and attributes. In addition, it is not straightforward how to balance the links and attributes information. In this article we describe these challenges and propose a fuzzy clustering model as well as a fuzzy clustering algorithm, HASCOP. Extensive experimentation on real world datasets has shown that HASCOP can be successfully applied in such networks, demonstrating its efficacy and superiority against the state-of-the-art attributed graph clustering methods.**

***Keywords*—*Clustering, Information Networks***

## I. Introduction

The proliferation of rich attribute information available for objects in real-world networks gives rise to *attributed graphs*. In an attributed graph, every vertex is characterized by a number of attributes describing its properties. The edges correspond to structural connections. Attributed graph as an expressive data structure is used in many application domains such as social networks, biology and telecommunications, representing both the graph topological structure and the vertex properties [1], [2], [6], [7].

Graph clustering is an interesting and challenging research problem which has received much attention recently [5]. However, in many real applications, both the graph topological structure and the vertex properties are important in clustering. For example, in a bibliography network, such as DBLP, a vertex may represent an author, vertex properties describe attributes of the author (such as the area of interest, number of publications, etc), while the topological structure represents relationships among authors (such as co-authorships, etc). Clustering the authors by considering both their personal profiles and the relationships among them is useful for example for researchers in identifying the most influential authors per area, recommending new collaborations, etc.

The problem we study in this paper is to efficiently cluster large-scale homogeneous information networks where vertices are connected by multiple types of links and characterized by different kinds of attributes, each of which has a different level of semantic importance. For example, in a social network, vertices may represent user profiles. In this context, user profile attributes may be name and location while different types of links represent relations (such as friendship, sharing a video etc), having different importance. Clustering this social network graph to find out the political trends, the attribute political views of a person is clearly more important than its name or gender. Similarly, a request to join a political group is more important than sharing a funny video. Furthermore, as two social network members may be connected by more than one link type, a social network is an *attributed multi-graph*.

Recently, there has been a great deal of research in the area of clustering attributed graphs [2], [6], [7]. These works focus on partitioning an attributed graph into dense clusters of vertices characterized by attribute homogeneity. However, none of these works has addressed the problem of detecting clusters of vertices in an attributed multi-graph, where links can be of different type and importance. Specifically, this article makes the following contributions:

- We propose HASCOP, a generic parameter-free attributed multi-graph clustering algorithm. HASCOP (Homogeneity Attributes and Similar COnnectivity Patterns) groups the vertices with similar connectivity and attributes into clusters that have high attribute homogeneity.
- We evaluate our method on a diverse collection of real data sets (bibliography items and software packages available on Google code repository). Experimental results show that HASCOP successfully groups vertices into meaningful clusters and reveals the outlier vertices. A performance evaluation of HASCOP against the state-of-the-art competitors is conducted, which attests its efficacy and superiority.

The rest of this article is organized as follows: Section II describes the related work. In Section III, the problem addressed in this paper is formally defined. Section IV presents the proposed clustering model and Section V describes the details of our clustering algorithm. Section VI provides the experimental evaluation of HASCOP and its comparison against the state-of-the-art competitors. Finally, Section VII concludes the present article.

## II. Related work

Graph clustering has been widely explored in the literature. A wide range of algorithms that have been proposed focus mainly on the connectivity structure of the graph [5]. The clustering results usually contain communities (strongly connected components) within clusters. However, graph clustering methods ignore vertex attributes and cannot be directly applied to attributed graphs. This section presents the related work and comparison. An overview of attributed graph clustering methods is depicted in Table I. Attributed graph clustering algorithms can be categorized into two types: Distance-based and Model-based.

**Distance-based Algorithms** use a similarity or distance measure that combines both structural and attribute information of the vertices. Based on this measure, traditional graph clustering algorithms are applied.

In this context, SA-Cluster [8] uses random walk distance in order to measure vertex closeness on an *augmented attributed graph*. An augmented graph is the initial graph enriched with new vertices that represent the attribute values. An edge from a graph vertex to an attribute vertex is added if the vertex is characterized by the attribute value represented by the attribute vertex. The weight of the new edge depends on the importance of the attribute the attribute vertex represents. By enriching the graph, the vertices which share the same attributes are closer and there is a path of at least two hops, through the attribute vertex, between them. Based on this distance measure, SA-Cluster takes a K-Medoids clustering approach to partition the graph. At each iteration the attribute weights are recalculated and the random walk distances on the augmented attributed graph are recalculated. To efficiently compute the random walk distances, authors proposed two extended versions: an incremental distance computation in Inc-Cluster [9] and an approximate distance computation in SA-Cluster-Opt [2].

**Model-based Algorithms** use probabilistic models to cluster attributed graphs. Generally a cluster is modelled by its connection and attribute distributions (e.g. exponential, Gaussian). These approaches calculate the parameters of the distributions, and a vertex is categorized in a cluster if its properties follow the cluster's distributions. BAGC [7] uses the Bayesian inference while GenClus [6] uses the expectation-maximization (EM) algorithm to partition the graph. Bayesian inference is an inference method in which Bayes' rule is used to update the probability estimate for a hypothesis as additional evidence is learned. EM is a general method of finding the maximum-likelihood estimate of the parameters of an underlying distribution from a given data set when the data is incomplete or has missing values. GenClus also adjusts link types weights based on the clustering configuration and can naturally be applied to attributed graphs with incomplete attributes.

**Discussion.** Clustering attributed graphs by applying the above approaches presents several non-trivial challenges. A key characteristic of all these approaches is that the number of clusters should be specified a priori. In addition, the use of augmented graphs (SA-Cluster) can firstly lead to an explosion of graph size, increasing significantly the requirements in memory and time; and secondly vertices which share very close, but not exactly the same, attributes are not getting closer. Furthermore, SA-Cluster and BAGC deal with only one type of links. On the other hand, although GenClus deals with different types of links, it considers all the attributes equally important and it is quite slow and hard to scale up because it is based on EM.

The work most similar to ours is PICS [1]. Similarly to HASCOP, PICS identifies clusters in an attributed graph with similar connectivity and attribute homogeneity. It considers two vertices to have similar connectivity if the sets of vertices they connect to highly overlap. Specifically, PICS compresses (conceptually, summarizes) the interrelated adjacency and attribute matrices simultaneously by splitting the clusters that require the most bits to be encoded. Similarly to HASCOP, PICS does not require any user-specified input such as the number of clusters. As opposed to our work, PICS splits clusters iteratively until the total encoding cost cannot be reduced any further. However, PICS does not consider neither the fact that attributes have different importance nor the different types of links that may exist in such networks.

Despite the success of the above methods, all of them lack one or more of the properties listed in Table I. It is noticeable that HASCOP is the only approach that deals with attributed multi-graphs and simultaneously automatically identifies the different importance of both link types and attributes.

## III. Problem Definition

In this section we introduce our notations, definitions and concepts. To make it easier to the reader, Table II presents the mathematical symbols that are used in the present article as well as a short description.

An attributed multi-graph is consisted of $|V|$ vertices described by $p$ attributes $\alpha_i, 1 \leq i \leq p$, and connected by $|E|$ directed edges. $E$ contains all the edges which are of $t$ different types. For simplicity, we limit the conversation only to unweighted connections. Such a network can be

TABLE II. Paper Notations

| Symbol | Description |
|---|---|
| $V$ | The set of vertices |
| $E$ | The set of edges |
| $p$ | The number of attributes/properties of each vertex |
| $A_{\|V\|\times p}$ | The attributes matrix |
| $\alpha_i$ | The $i^{th}$ attribute. $1 \leq i \leq p$ |
| $domain(\alpha_i)$ | The domain of $\alpha_i$ |
| $L^i_{\|V\|\times\|V\|}$ | The link matrix for links of type $t_i$ |
| $t$ | Total number of link types |
| $k$ | The number of clusters |
| $\Theta_{\|V\|\times k}$ | The fuzzy clustering configuration |
| $c_i$ | The $i^{th}$ cluster |
| $S_j$ | The set of vertices that belong to $c_j$. $S_j \subseteq S(c_j) \subseteq V$ |
| $S(v_i)$ | The set of vertices $v_i$ connects to and $v_i$ itself |
| $S(c_j)$ | The set of vertices in $S_j$ and the vertices they link to. |
| $\mathcal{C}^{attr}_{k\times p}$ | The centroids of the clusters based on attributes |
| $\mathcal{C}^{links}_{k\times\|V\|}$ | The link properties of the clusters |

TABLE I. Comparison of Related Work

| | Directed | Weighted | Multi-graph | Attribute weights | Link-type weights | Similar Connectivity | Parameter free |
|---|---|---|---|---|---|---|---|
| SA-Cluster [8] | ✓ | ✓ | | ✓ | | | |
| BAGC [7] | ✓ | | | | | | |
| GenClus [6] | ✓ | ✓ | | | ✓ | | |
| PICS [1] | ✓ | | | | | ✓ | ✓ |
| HASCOP | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ |

represented by $t+1$ matrices: $A_{|V|\times p}$ and $L^i_{|V|\times|V|}$, where $1 \leq i \leq t$. Matrix $A$ represents the attributes where the $i^{th}$ row, $A_i$, is the attributes vector of vertex $v_i$. $A_{i,j} = x$ means vertex $v_i$ at attribute $\alpha_j$ has the value $x$. The importance of attribute $\alpha_i$ is represented by the $i^{th}$ element, $w_{\alpha_i}$, of the $p \times 1$ vector $\vec{w_\alpha}$. Matrix $L^i$ is the adjacency matrix for the links of type $l_i$. An edge $(u, v, l_i)$ belongs to $E$ if there is a connection from vertex $u$ to vertex $v$ of link type $l_i$. Link type $l_i$ also has a weight, $\omega_{l_i}$, which is the $i^{th}$ element of the $t \times 1$ vector $\vec{\omega_l}$, based on its semantic meaning. The elements of the weight vectors, $\vec{\omega_l}$ and $\vec{w_\alpha}$, must be learned according to the importance of each link type and attribute.

Our goal is to assign each vertex $v_i$ a vector $\theta_{k\times 1}$, where $k$ is the number of clusters and $\theta_j$ is the probability of $v_i$ belonging to cluster $c_j$. Vertices in the same cluster should exhibit **both** *similar connectivity pattern* and *attribute coherence*. In the following paragraphs we give an overview of the similar connectivity and attribute coherence.

Two vertices $v_i$, $v_j$ have similar connectivity pattern if $S(v_i)$ and $S(v_j)$ highly overlap, where $S(v_i)$ is the set containing $v_i$ itself and the vertices that $v_i$ connects to. If $S(v_i)$ and $S(v_j)$ highly overlap then $v_i$ and $v_j$ link to common vertices. It is noted that vertices having similar connectivity, may not be directly connected with each other. Thus, the density metric of the final clustering may not be close to one because clusters are not necessary densely connected components. The intuition is that vertices related to common vertices should form a cluster even though they are not inter-connected. For instance, in the case of a social network, if some people have a lot of common friends (and common attributes) but they do not know each other, they should form a group, and such groups are covered as well. Furthermore, it is desired that the similarity connectivity pattern of two vertices that link to common vertices to be higher if they are also connected to each other. For example, a group of people having common friends should belong to the same clusters with higher probability if they are also friends with each other. To address this issue, we perform an augmentation of the original graph connecting each vertex to itself with $t$ typed links. Hence, $v_i$ belongs to $S(v_i)$.

Two vertices $v_i$, $v_j$ have attribute coherence if $A_i$ and $A_j$ are close in $\mathcal{R}^p$. The intuition is that vertices described by close attributes should have high attribute coherence.

Formally, given the matrices $A_{|V|\times p}$ and $L^i_{|V|\times|V|}$, the goals are:

- Identify the importance of each link type and calculate their weights $\omega_{l_i}$, where $\omega_{l_i} \in [0,1]$ and $\sum_{i=1}^{t} \omega_{l_i} = 1$, based on the clustering configuration and the structural properties of the graph.
- Give each attribute a weight $w_{\alpha_i}$ based on its importance, where $w_{\alpha_i} \in [0,1]$ and $\sum_{i=1}^{p} w_{\alpha_i} = 1$.
- Find a fuzzy clustering configuration $\Theta_{|V|\times k}$, where $k$ is the number of clusters, $\Theta_{i,j}$ is the probability of $v_i$ belonging to cluster $c_j$ and $\sum_{i=0}^{k} \Theta_{x,i} = 1 \; \forall \, x \in [1, |V|]$.

## IV. HASCOP Clustering Model

### A. Overview

HASCOP fuzzy clustering model for attributed multi-graphs makes use of a similarity function $s$ that combines both their link and attribute properties according to the weight vectors $\vec{\omega_l}$ and $\vec{w_\alpha}$. Hence, we want to maximize the following objective function:

$$O(\Theta, \vec{\omega_l}, \vec{w_\alpha}) = \sum_{i=1}^{|V|} \sum_{j=1}^{k} \Theta_{i,j} \cdot s(v_i, c_j, \vec{\omega_l}, \vec{w_a}) \qquad (1)$$

where $s(v_i, c_j, \vec{\omega_l}, \vec{w_a})$ is the similarity of vertex $v_i$ and cluster $c_j$.

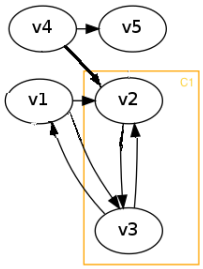
HASCOP maximizes the above function as follows. Initially each vertex is categorized in a cluster by itself. Iteratively clusters are being updated (some are eliminated) and link-type and attribute weights are adjusted. An iteration starts with categorizing the vertices into the updated clusters of the previous iteration, according to their similarity and the updated weights. A link type or an attribute is considered more important and is given higher weight in the next iteration if vertices in current clusters connect to same vertices by this link type or share the same value for the specific attribute respectively. The $i^{th}$ rows of two interrelated matrices, $\mathcal{C}^{links}_{k\times|V|}$ and $\mathcal{C}^{attr}_{k\times p}$, represent the link and attribute properties of cluster $c_i$ respectively.

### B. Similar Connectivity

As stated earlier, vertices can be connected via $t$ different types of links. The elements of matrix $L^k$, $k \in [1, t]$, represent the edges of link type $l_k$. The complete set of edges E can be represented as an aggregated adjacency matrix, $L_{|V|\times|V|}$, which is defined as the weighted sum of matrices $L^k$, based on the different importance of the various link types.

Given the matrices $L$ and $\mathcal{C}^{links}$ we must define a similarity function $link\_sim(v_i, c_j)$ which returns the similar connectivity pattern of $v_i$ and $c_j$. Function $link\_sim$ takes a value in the range $(0, 1]$ and is given by:

$$link\_sim(v_i, c_j) = \frac{1}{1 + \sqrt{\sum_{x=1}^{|V|} \left(L_{i,x} - \mathcal{C}^{links}_{j,x}\right)^2}} \qquad (2)$$




| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ |
|---|---|---|---|---|---|
| $C_1$ | 1 | 1 | 1 | 0 | 0 |
| $L_1$ | 1 | 1 | 1 | 0 | 0 |
| $L_2$ | 0 | 1 | 1 | 0 | 0 |
| $L_3$ | 1 | 1 | 1 | 0 | 0 |
| $L_4$ | 0 | 1 | 0 | 1 | 1 |
| $L_5$ | 0 | 0 | 0 | 0 | 1 |

Fig. 1. Similarity Connectivity

If $L_i$ is the same with $\mathcal{C}_j^{links}$, meaning that $v_i$ connects to the same vertices as $c_j$, then $link\_sim(L_i, \mathcal{C}_j^{links}) = 1$ because their distance is zero. On the opposite side, if there is not a vertex $v_x$ that both $v_i$ and $c_j$ connect to then their distance will be high and their similar connectivity will be very close to zero. The value one is added to the distance to ensure that the result is in the range $(0, 1]$ as the distance can be less than one. For example, for the graph shown in Figure 1, $link\_sim(v_1, c_1) = 1.0$ since $L_1$ and $\mathcal{C}_1^{links}$ are exactly the same. On the other hand, $link\_sim(v_5, c_1) = \frac{1}{3}$ and $v_5$ should not be placed into $c_1$. Equation 2 is preferred over any other traditional measures as it takes into account the outgoing links of a vertex and not its adjacent vertices.

### C. Attributes Coherence

We calculate the attribute similarity of $v_i$ and $c_j$, based on the attributes importance, by transforming to similarity their weighted Euclidean distance using the following monotonically decreasing transformation:

$$attr\_sim(v_i, c_j, \vec{w_\alpha}) = \frac{1}{1 + \sqrt{\sum_{l=1}^{p} w_{\alpha_l} \cdot \left(A_{i,l} - \mathcal{C}_{j,l}^{attr}\right)^2}} \tag{3}$$

We add one to the denominator for two reasons. The first reason is to ensure that $attr\_sim$ is always in the range $(0, 1]$, and the second reason is to ensure that result is close to one if the attribute vector of $v_i$ is very close to the attribute centroid of $c_j$.

### D. Similarity Function and Membership Calculation

Based on Equations 2, 3 we now define the function $s(c_j, v_i, \vec{w_a})$ of the objective function as follows:

$$s(c_j, v_i, \vec{w_a}) = link\_sim(v_i, c_j) \cdot attr\_sim(v_i, c_j, \vec{w_\alpha}) \tag{4}$$

The reason for selecting to multiply the two similarities is simple; a vertex has high similarity with a cluster if **both** their similar connectivity and attribute coherence are high.

Finally, we calculate memberships using the following equation:

$$\Theta_{i,j} = \frac{s(c_j, v_i, \vec{w_a})}{\sum_{\forall k} s(c_k, v_i, \vec{w_a})} \tag{5}$$

Equation 5 assigns a vertex in a cluster with high probability if their similarity is high and the objective function is maximized.

### E. Links and Attributes Weights Adjustment

Given the matrices $\Theta$, $\mathcal{C}^{links}$ and $\mathcal{C}^{attr}$ we must adjust the weights, $w_{a_i}$ and $\omega_{l_i}$, for each attribute, $\alpha_i$, and link type, $l_i$. In order to automatically adjust the weights we use a voting mechanism similar to the one described in [8]. Since HASCOP is iteratively clustering the network, before proceeding to the next iteration the link type and attribute weights are adjusted according to their score. On each iteration a score is assigned to each link type and attribute based on the global updated clustering configuration $\Theta$ and their importance. A link type or an attribute is considered more important if vertices in the same cluster are interconnected by this link type or share the same value for the specific attribute respectively.

Let $\omega_{l_i}^r$ be the weight of link type $l_i$ at iteration $r$. Before proceeding to the next iteration $\omega_{l_i}^{r+1}$ must be calculated. Based on $\Theta$ and the importance of link type $l_i$, a score $\Delta\omega_{l_i}^r$ is calculated, such that $\sum_{i=1}^{t} \Delta\omega_{l_i}^r = 1$. The updated weight for link type $l_i$ is then given by the average of $\omega_{l_i}^r$ and $\Delta\omega_{l_i}^r$. Because $\sum_{i=1}^{t} \omega_{l_i}^r = 1$ and $\sum_{i=1}^{t} \Delta\omega_{l_i}^r = 1$ the constraint $\sum_{i=1}^{t} \omega_{l_i}^{r+1} = 1$ is not violated. Obviously, if a link type $l_i$ is more important it will get higher weight in the next iteration if it has $\Delta\omega_{l_i}^r > \omega_{l_i}^r$. On the other hand, if is not that important then $\Delta\omega_{l_i}^r$ must be less than $\omega_{l_i}^r$. We define the function $link\_score\left(L^i, \Theta\right)$ which returns the score of link type $l_i$ based on the clustering configuration. The function $link\_score$ is defined as:

$$link\_score_i\left(L^i, \Theta\right) = \sum_{a=1}^{|V|} \sum_{b=1, b\neq a}^{|V|} \sum_{j=1}^{k} \left(\Theta_{a,j} + \Theta_{b,j}\right) \tag{6}$$
$$\text{where } L_{a,b}^i \neq 0, \Theta_{a,j} \neq 0, \Theta_{b,j} \neq 0$$

In other words, to calculate the $link\_score_i$, we add the probabilities of each pair of vertices $v_a$ and $v_b$ that belong to a cluster $c_j$, if there is a link of type $l_i$ from $v_a$ to $v_b$. It is noted that $link\_score_i$ value is not a probability but a measure of the importance of links of type $l_i$ in the current clustering. The higher the $link\_score_i$ the more important the link type $l_i$ is. Similarly to links, the higher the score of an attribute the more important it is. For an attribute its score is given by the sum of the number of vertices in the same cluster that are sharing the same value for that attribute.

Having calculated all scores, $\Delta$ for link types and attributes are given by:

$$\Delta\omega_{l_i}^r = \frac{link\_score_i}{\sum_{l=1}^{t} link\_score_l}, \Delta w_{\alpha_i}^r = \frac{attr\_score_i}{\sum_{l=1}^{p} attr\_score_l} \tag{7}$$

**Algorithm 1** HASCOP

**Input:** Link matrices $L^i_{|V|\times|V|}$, Attribute matrix $A_{|V|\times p}$
**Output:** Fuzzy clustering configuration $\Theta_{|V|\times k}$
Initialize weights
2: Calculate aggregated matrix $L$
$\mathcal{C}^{links} \leftarrow L$ , $\mathcal{C}^{attr} \leftarrow A$
4: **while** true **do**
**for** $i = 1 \rightarrow |V|$ **do in parallel**
6: **for** $j = 1 \rightarrow k$ **do**
Calculate $\Theta_{i,j}$ according to Equation (5)
8: **end for**
**if** $\Theta_i$ is empty **then**
10: $\Theta_i \Leftarrow \Theta_i^{r-1}$
**end if**
12: **end for**
Parallel update $\mathcal{C}^{links}$ and $\mathcal{C}^{attr}$
14: Parallel update all weights
**if** NOT ProcessDuplicatesClusters($\Theta$) **then**
16: return $\Theta$;
**end if**
18: **end while**

## V. HASCOP Algorithm

Following paragraphs describe the initialization and the clustering process of HASCOP as well as the way the clusters are handled and updated on each iteration.

Initially, if there is not previous knowledge on the importance of each link type and attribute, their weights are set equal and given by: $\omega_{l_i} = \frac{1}{t}$, $w_{a_i} = \frac{1}{p}$. For initialization of clusters we set $\mathcal{C}^{links} = L$ and $\mathcal{C}^{attr} = A$ meaning that each cluster is consisted by only one vertex.

After the calculation of $\Theta$, it is not guaranteed that clusters containing exactly the same vertices do not exist. Thus, a process that eliminates the duplicate clusters must be performed. Specifically, the clusters which contain exactly the same vertices are all deleted except one.

At the end of each iteration, the clusters are updated according to the new clustering configuration $\Theta$. The elements of $\mathcal{C}^{links}_j$ and $\mathcal{C}^{attr}_j$ are given, similarly to fuzzy k-means algorithm, as the weighted average of the characteristics of vertices in cluster $c_j$.

HASCOP takes as parameters only the matrices $L^i$ and $A$, which represent the links and attributes of the network. The pseudo-code for the clustering process is presented by Algorithms 1. HASCOP converges when at the end of an iteration the number of clusters does not change.

After initialization (lines 1-3), the first step is the calculation of $\Theta$ according to Equation (5) as shown in lines 5-11. As soon as the temporary $\Theta$ is calculated, the clusters and the link type and attribute weights are updated (lines 13-14). Finally, the last step, which determines if the algorithm has converged at this iteration, is the processing of same clusters. Due to the independence of values being updated simultaneously, HASCOP has been *implemented and executed in parallel*. Such a parallel implementation confirms the scalability of the proposed algorithm.

### A. Time Complexity

The time complexity of HASCOP can be expressed as the sum of the costs for calculating $\Theta$, processing duplicate clusters, updating centroids and updating link-type and attribute weights.

The time complexity for calculating $\Theta_{i,j}$ is the cost for calculating the link and attribute distances between vertex $v_i$ and cluster $c_j$. The cost for calculating the weighted Euclidean distance of two vectors $v_{d\times 1}$ is $O(d)$. Therefore, the cost for calculating the attribute distance is $O(p)$. Link vectors are sparse and using appropriate data structures the cost for calculating the link similarity is $O(\frac{|E|}{|V|})$, assuming the average size of $S(v_i)$ is $\frac{|E|}{|V|}$ (non zero elements in a link vector). Thus, the total cost for calculating a similarity is $O(\frac{|E|}{|V|} + p)$. Since the columns of $\Theta$ are $k_i$, where $k_i$ is the number of clusters at the end of the first step at iteration $i$, the cost for calculating $\Theta$, lines 5-12, is $O(|V| \cdot k_i \cdot (p + \frac{|E|}{|V|}))$. The time complexity for deleting the same clusters is $O(k_i)$ and the cost for updating clusters based on $\Theta$ is $O(k_i \cdot |V| \cdot (t+p))$. The time complexities for updating link type and attribute weights are $\approx O(t \cdot |V|)$ and $\approx O(p \cdot |V|)$ respectively. Thus, the total cost for updating all weights (line 14) is $\approx O(|V| \cdot (t+p))$.

Putting them all together, the total time complexity, for $r$ iterations, is $\approx O(r \cdot k \cdot \left[|V| \cdot \left(\frac{t+p}{k}\right) + |E|\right])$. In other words, the time complexity is approximately linear in the size of the graph and the properties of the vertices. Therefore, HASCOP algorithm is quite scalable.

TABLE III. Datasets

| Dataset | $\|V\|$ | $\|E\|$ | $p$ | $t$ | Directed |
|---|---|---|---|---|---|
| **DBLP-1000** | 1000 | 17128 | 2 | 1 | No |
| **GoogleSP-23** | 1297 | 24153 | 5 | 2 | No |

## VI. Experimental Study

In order to evaluate the proposed approach, HASCOP has been implemented in java 1.6 and compared to the the state-of-the-art competitors, SA-Cluster [2], [8], BAGC [7], GenClus [6] and PICS [1]. For HASCOP and GenClus, which perform fuzzy clustering, each vertex was considered to belong to the cluster with the highest probability for all experiments. Thus, the number of clusters returned by GenClus may be less than the predefined $k$. Experiments executed on a Dell Server equipped with two Intel Xeon 3.47GHz CPUs, 12 cores per CPU and 80GB RAM.

### A. Datasets

Table III summarizes the properties of the datasets information networks used in our experiments, showing the number of vertices, edges, attributes, link types and if the graph is directed.

*1) DBLP Bibliography Dataset (DBLP-1000):* DBLP-1000 dataset is a subset of the complete DBLP dataset[1] and contains 1000 vertices which represent the top authors from the complete DBLP dataset. Each vertex represents an author described by two attributes, the first attribute is the number of publications and the second is the primary topic of interest of the author. There are four topics which are databases (DB), data mining (DM), information retrieval (IR) and artificial intelligence (AI). An edge $(v_i, v_j)$

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

TABLE IV. FILE ATTRIBUTES AND THEIR DESCRIPTION

| Attribute | Description |
|---|---|
| FILE_SIZE | The size of the file in bytes |
| A_TIME | Last access time in seconds since Epoch |
| M_TIME | Last content modify time in seconds since Epoch |
| C_TIME | Time of most recent metadata change on Unix, or the time of creation on Windows in seconds |
| FILE_TYPE | The type of the file e.g. text/x-java, application/x-executable |

represents that authors with ids $i$ and $j$ have co-authored at least one publication.

*2) Software Packages (GoogleSP-23):* GoogleSP-23 is a dataset constructed from 23 software packages that were downloaded from the google code repository[2]. The software packages were installed into virtual machines on Nephelae cloud infrastructure[3]. The advantage of this dataset is that the clusters are known (software packages). Thus, all the algorithms can be compared to the ground truth. Due to space limitation, the description of the process of building the GoogleSP-23 dataset is omitted since it is out the scope of this work.

In GoogleSP-23 dataset, a vertex represents a file described by the attributes shown in Table IV. There are two types of links in this dataset, one for the file name similarities and one for the path similarities. A fictitious algorithm that has identified all the software packages (also displayed as "Optimal" in the following paragraphs to make comparison easier) results in density value of $0,0715$. Although this confirms that software packages are not densely connected components [4], SA-Cluster, BAGC and GenClus fail to identify any software packages. Thus, we transofrmed this dataset such that a software package is a densely connected component and should be identified by these approaches. The second version of this dataset contains an edge between $file_i$ and $file_j$ if both files belong to the same software package. Files are described by the same attributes.

## B. Evaluation Measures

This subsection presents the evaluation metrics used in all the experiments. As HASCOP returns a fuzzy clustering configuration, in conjunction to SA-Cluster, BAGC and PICS which produce a hard clustering, the fuzzy clustering configurations must be converted as described earlier. By this conversion, the same metrics, which are the entropy and the density, can be used for all the algorithms. Low entropy is equivalent to high homogeneity between the attributes of the vertices in the same cluster. High density value means higher connectivity between vertices in the same cluster.

Furthermore, for the evaluation of the results for the Google-SP23 dataset we use the Dice similarity score. Dice score takes a value in the range of $[0, 1]$, and the higher the score the more the two sets are similar. We use it to measure how many of the 23 software packages have high dice score with at least one cluster. Moreover, we want to see how well the software files are separated into clusters. Thus, we want a metric that tell us how many of the files in a cluster are from the same software package. For this purpose we adopt the following score:

$$Score(A, B) = \frac{|A \cap B|}{|A|} \quad (8)$$

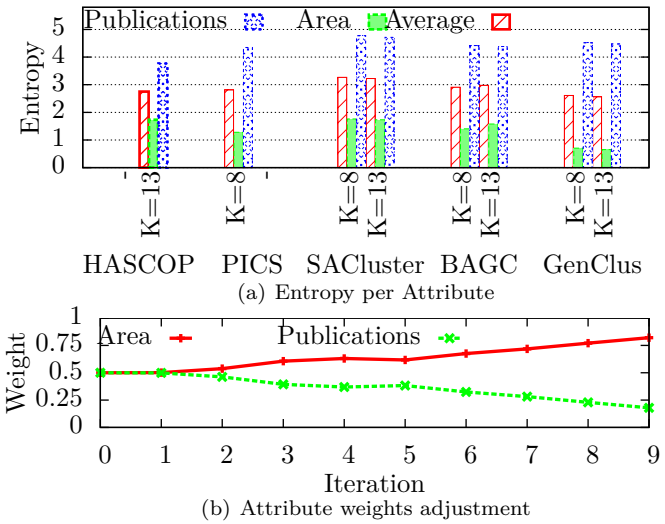
If all the files in cluster $A$ are part of the software package $B$ then $score(A, B) = 1.0$, on the other hand, if no file in $A$ belongs to $B$ then $score(A, B) = 0.0$. The above score is used due to the fact that usually a software package is consisted by more than one software components and HASCOP in some of these cases identified the software components instead of the complete software package.

## C. Evaluation

*1) Clustering the DBLP-1000 dataset:* For the DBLP-1000 dataset, HASCOP and PICS returned 13 and 8 clusters respectively. BAGC and SA-Cluster were also executed for k=8 and k=13. GenClus resulted in 4 clusters for both k, as each author was placed only in the cluster with the highest probability.

Figure 2(a) shows the entropy for both attributes in this dataset (area and number of publications). As expected, the entropy for the number of publications is higher due to the fact that authors rarely have exactly the same number of publications. For the area attribute the entropy is lower, as a lot of authors are interested in one area, but is not zero for none of the algorithms. It is shown that HASCOP clustering is described by one of the lowest average entropy values, confirming that HASCOP succeeds in identifying clusters with attribute homogeneity.

Clustering of the DBLP-1000 dataset can verify the correctness of the weight update mechanism. Figure 2(b)




(a) Entropy per Attribute

(b) Attribute weights adjustment

Fig. 2. Clustering the DBLP-1000 dataset

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

[3]Provided by LINC, University of Cyprus http://grid.ucy.ac.cy

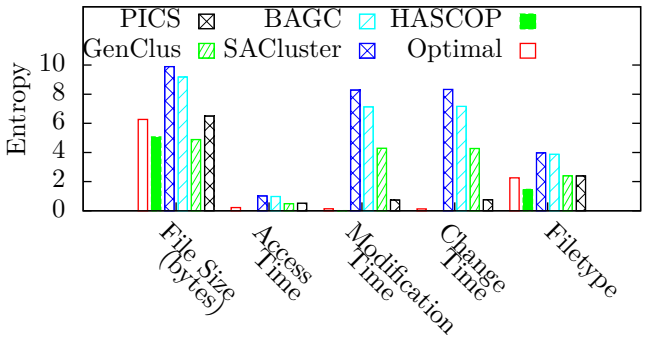


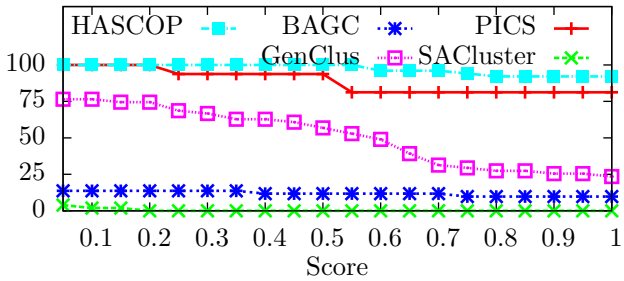

Fig. 3. Entropy per attribute (GoogleSP-23)

shows the weight adjustment process for all the iterations needed to cluster this dataset. As clustering goes on, as expected, area of study gets significantly higher weight than the number of publications after the first two iterations. This happens because during the first two iterations authors having much more publications than others are not characterized as outliers, resulting in lower score for area attribute. Furthermore, during the first two iterations authors were not placed into groups based on their area of study, but grouped into clusters with almost same number of publications. Finaly, higher entropy of publications attribute lead to its weight decrease. Figure 2(b) confirms that the weight update mechanism has correctly identified the importance of each attribute as without doubt area of interest is more important than the number of publications.

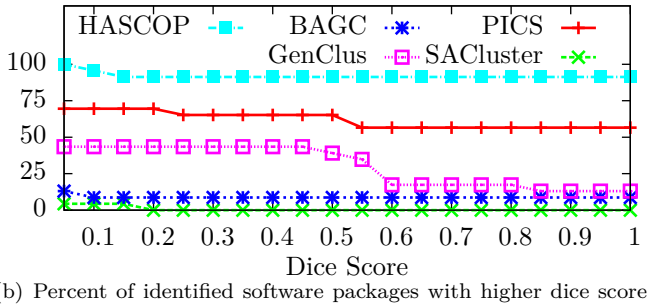
Inspecting and interpreting the final clusters we see that HASCOP has successfully revealed important characteristics of the DBLP-1000 dataset. HASCOP identified and categorized separately three authors, because they have the highest number of publications. For the first category there were two clusters with the authors with the most publications. These authors were placed into two clusters due to the different number of co-authorships. Three clusters were containing the authors with the most publications for each of the other categories, and another cluster contained all the authors with the least publications from all categories. The rest four clusters contained the authors for each area, having the most co-authorships. This results can be of great help to researchers analysing the DBLP or similar datasets.

*2) Clustering the GoogleSP-23 dataset:* For the purpose of identifying software packages the constructed dataset was used. In this experiment PICS returned only 16 clusters, and SA-Cluster, BAGC and GenClus were executed only with k=51, which is the number of clusters returned by HASCOP. GenClus resulted in 39 clusters after keeping each file in the cluster it belongs with the highest probability. The number of clusters is higher than the number of the software packages, as a software package is usually consisted of more than one software component.

Figure 3 shows the entropy per attribute for the resulted clustering configurations. "Optimal" refers to the fictitious algorithm that has identified all the software packages (ground truth). Entropy must be as close as possible to




(a) Percent of clusters overlapping with software packages with higher overlap score in horizontal axis



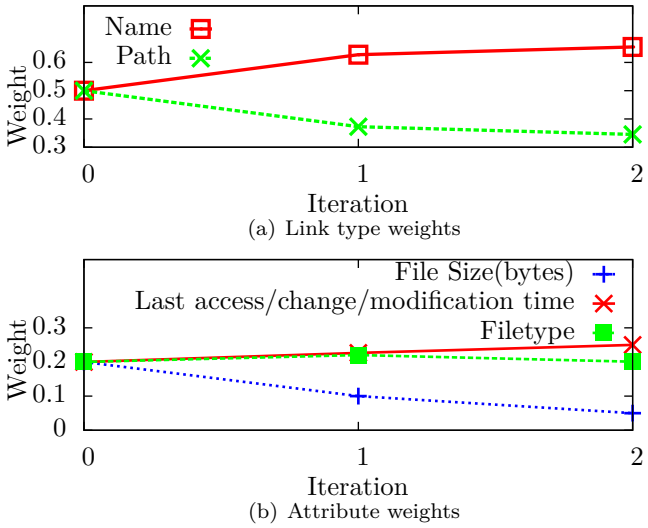

(b) Percent of identified software packages with higher dice score in horizontal axis

Fig. 4. Clustering the GoogleSP-23 dataset

the "Optimal". It is shown that HASCOP is the closest to the "Optimal" and in addition succeeds to have the lowest entropy for all attributes. Lowest entropy for all the attributes confirms that HASCOP successfully identifies clusters characterized by high attribute homogeneity.

In order to give meaning to the results, Figure 4(a) shows how many clusters overlap with software packages. The vertical axis shows the percentage of clusters with overlap score, using Equation 8, with any software package higher than the value shown in the horizontal axis. It is noticeable that more than 80% of returned clusters by HASCOP and PICS have score 1.0, which means all files in these clusters are from the same software packages. Equally, if we select a random cluster returned by HASCOP or PICS, that cluster contains files only from one software package with probability higher than 0.9 or 0.8 respectively. Figure 4(a) shows that HASCOP and PICS separated software files in different clusters with higher precision that the other algorithms.

Figure 4(b) shows the percentage of software packages that were successfully completely identified. That is, the percentage of software packages that have higher dice score than the value in x-axis with at least one cluster. As shown from both figures, SA-Cluster, BAGC and GenClus fail completely to identify the software packages, in conjunction to PICS which returned 16 clusters fully overlapping with 13 software packages. On the other hand, almost all clusters (~90%) returned by HASCOP have full overlap with a software package and almost all software packages (21 out of 23) have been successfully identified. In other words, a randomly selected cluster returned by HASCOP fully represents a software package with probability higher than 0.9. It is also noticed that SA-Cluster, BAGC and




(a) Link type weights

(b) Attribute weights

Fig. 5. Weights adjustment (GoogleSP-23)

GenClus failed to identify the software packages, even though in the second version of the dataset, a software package is densely connected. In this experiment, the importance and usability of similar connectivity patterns and attribute homogeneity is also confirmed.

Furthermore, Figures 5(a) and 5(b) show the weight adjustment process for both link types and attributes respectively as clustering goes on. It is mentionable the fact that path similarity is getting lower weight as clustering converges, revealing that it would be totally wrong to categorize files in software packages by just viewing their path. On the other hand, the name similarity receives more importance. This is due to the fact that usually files in the same software package have similar names, e.g. chat.css, chatEditor.js, chatButton.js etc.. For the attributes, higher importance is given to the time of last access, last modification and last change in conjunction to file size and file type. This was expected as a software package contains files of different sizes and file types, while files belonging to same software packages have similar access and modification timestamps. Figure 5 attests the correctness and necessity of the weight update mechanism in order to get meaningful results.

*Summarizing the results:* HASCOP succeeded in returning meaningful clusters for the used datasets. Clusters returned by HASCOP, containing vertices with similar connectivity patterns and attribute homogeneity, can be useful to many applications dealing with such information networks. Furthermore, entropy was low for all experiments revealing that HASCOP can successfully identify clusters characterized by attribute homogeneity. Looking at Figures 2(b) and 5, we conclude that the importance of each link type and attribute can be successfully identified. Based on the weight update mechanism and the presented similarity function HASCOP can reveal important properties of the attributed multi-graphs under study.

## VII. Conclusions and Future work

In this paper, a generic approach for clustering attributed multi graphs is presented. The proposed method, HASCOP, is consisted of two main steps, the assignment of vertices into clusters and the adjustment of the link type and attribute weights. Experiments on real world datasets have shown its adaptiveness and superiority against the state-of-the-art competitors. The weight adjustment process can successfully identify the importance of each link type and each attribute giving each of them the appropriate weight. Furthermore, the importance of similar connectivity patterns is also addressed and analysed in the present article giving a boost to new research studies of such complex information networks.

HASCOP can be successfully applied to a wide range of homogeneous information networks such as the DBLP-1000 and GoogleSP-23. Clustering such networks into clusters with attribute homogeneity and similar connectivity patterns returns meaningful clusters and reveals important characteristics of the dataset under study. Furthermore, in contrast to other algorithms, HASCOP has correctly identified the software packages installed on a cloud computing infrastructure. As an extension of this successful method, the next step is its integration into Minersoft search engine [3] for software retrieval in clouds.

We furthermore pose our future directions to the extension of this powerful approach to weighted multi-graphs where vertices can be of different types as well as the deployment to a large scale hadoop cluster to process large heterogeneous information networks.

**Acknowledgement.** The authors would like to thank the authors of SA-Cluster, BAGC, GenClus and PICS for their cooperation. We also thank Dr. Katakis I. for his valuable comments.

## References

[1] L. Akoglu, H. Tong, B. Meeder, and C. Faloutsos, "PICS: Parameter-free Identification of Cohesive Subgroups in Large Attributed Graphs," in *SDM*, Anaheim, CA, 2012.

[2] H. Cheng, Y. Zhou, and J. X. Yu, "Clustering Large Attributed Graphs: A Balance between Structural and Attribute Similarities," *ACM Trans. Knowl. Discov. Data*, vol. 5, no. 2, Feb. 2011.

[3] M. D. Dikaiakos, A. Katsifodimos, and G. Pallis, "Minersoft: Software retrieval in grid and cloud computing infrastructures," *ACM Trans. Internet Technol.*, vol. 12, no. 1, Jul. 2012.

[4] S. Jenkins and S. Kirk, "Software architecture graphs as complex networks: A novel partitioning scheme to measure stability and evolution," *Information Sciences*, vol. 177, no. 12, 2007.

[5] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, 2007.

[6] Y. Sun, C. C. Aggarwal, and J. Han, "Relation strength-aware clustering of heterogeneous information networks with incomplete attributes," *Proc. VLDB Endow.*, vol. 5, no. 5, Jan. 2012.

[7] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng, "A model-based approach to attributed graph clustering," in *Proceedings of the 2012 international conference on Management of Data*, ser. SIGMOD '12. New York, NY, USA: ACM, 2012.

[8] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," *Proc. VLDB Endow.*, vol. 2, no. 1, Aug. 2009.

[9] Y. Zhou, H. Cheng, and J. Yu, "Clustering large attributed graphs: An efficient incremental approach," in *Data Mining (ICDM), IEEE 10th International Conference on*, Dec. 2010.