

# Information-theoretic Detection of Non-Redundant Overlapping Communities in Attributed Graphs

Jing Feng

Nina Hubig

Xiao He

Claudia Plant

integrative Knowledge Discovery in Databases Group

University of Munich, Helmholtz Zentrum München, Technische Universität München

{feng,he}@dbs.ifi.lmu.de, {nina.hubig, claudia.plant}@helmholtz-muenchen.de

## ABSTRACT

How to mine complex and useful knowledge from an attributed graph? For many real world data, consider e.g. social networks, we do not only have a graph representing e.g. the friendship network, but also a lot of additional attributes further characterizing the nodes, e.g. information on hobbies, gender, location, etc. Some techniques for clustering such complex data have recently been proposed. However, existing algorithms suffer the drawbacks that they either require many input parameters which are difficult to estimate, or they only recognize the overlap in the graph structure or in the attributes of the nodes but not in both. With overlap we mean that nodes as well as attributes might be assigned to multiple different clusters. Avoiding overlap implies losing information. For that reason we propose a method that enables combined overlap in both the underlying network structure and the attribute subspaces of each node. But allowing that much overlap raises another challenge: How to handle redundancy? As solution this paper proposes a non-redundant method called IROC (for *Information-theoretic non-Redundant Overlapping Clustering*) that evaluates complex attributed graphs in an automatic way. We are the first that emphasize the importance of mining overlapping communities in all aspects of an attributed graph. Our experiments show that IROC reveals meaningful information in both real world - and synthetic data sets.

## Keywords

graph mining, social networks, attributed graph,

## 1. INTRODUCTION

Social networks, gene-and/or protein interaction networks as well as nearly all common types of networks in real world applications are not only sharing a large amount of information depending on the relationship between the vertices, but also contribute information regarding the characteristics of these vertices. These characteristics are modeled as attributes of a graphs' vertex. Thus we are referring to a

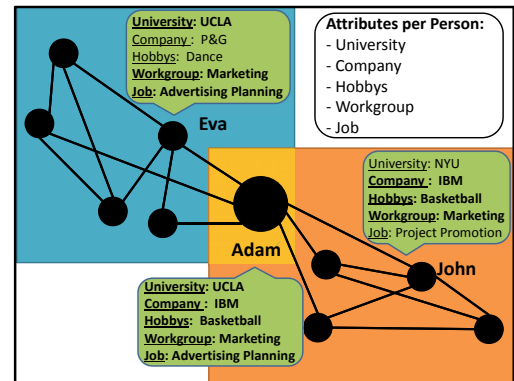


Figure 1: Motivational example of a social friendship network.

graph containing extra information in its nodes as an *attributed graph*. These attributed graphs connect two aspects of information: First, the structural "who knows whom" connection given by the graph itself as for example in friendship relationships of applications like facebook. Second, the attributes per node or person showing (in the case of facebook) personal ego-information like hobbies, what they are working, where they are living etc. Combining both aspects of information gives the chance to analyze things like "what has this friendship circle in common" or "what do most people in my work environment like and what joins them together?". Questions like this can be answered to some extend by just clustering attributed graphs, as existing algorithms propose in [19], [16] and [18]. But we are not only clustering attributed graphs but are also applying an overlapping concept to both informational aspects - to the graph structure and to the attribute space. What do we mean by overlapping and what do we gain out of the attributed graph if we are already able to group both informational aspects by just partitioning the attributed graph?

Consider the example given in Fig.1, it outlines the meaning of "overlapping" in combination with an attributed graph. The figure shows the friendship circles in a social network of a person called "Adam". His two friendship circles are visualized in colored rectangles behind the graph structure: blue stands for his friends from school and orange indicates his circle of colleagues. The overlapping of both circles is highlighted in yellow. Every person (node) includes attributes on their university, their working place and department, what hobbies they have and what job position they are in. You

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WOODSTOCK '97 El Paso, Texas USA

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$15.00.

could also think of these attributes as more of a list, containing several items and differing in detail. For example, Adam graduated from UCLA, works in IBM, likes playing basketball, works in the marketing department and is mainly in charge of planning advertisements. Now, how would an attributed graph partitioner most likely divide this structure if no overlap is allowed? A distinct possibility is to assign Adam to either the blue or the orange circle, because Adam can form a nearly full clique with both circles. This would result in a very high quality for clustering for one cluster while the other would lose some information (Adam). In this frequent case, partitioning must sacrifice the quality of this one cluster. Obviously, overlapping clustering, like assigning Adam to both circles, provides more meaningful structural information. Lets take a look at the two named nodes John and Eva that share some overlap in their attributes with Adam. John is a working partner of Adam, and shares the same hobby, workgroup and of course company with him. Also Eva graduated from the same university, shares the same workgroup and is working in a similar job like Adam, just in a different company. Therefore, Adam shares similar subspace attributes with his work circle where John is, while he shares other subspace with his school circle where Eva is. Apparently Adam should be assigned to both circles from the attribute information side as well. All in all this small example already shows the range of interpret-ability and complexity reduced from real world data when overlapping is not integrated.

Therefore, we contribute a new method of clustering attributed graphs with the objective to find the reasonable overlapping communities and meaningful subspace of the attributes at the same time based on an information theoretical technique. The advantages of the proposed algorithm are in short:

- **Discovery of overlapping communities:** Our method discovers not only single dense groups of friends but also their connections to other groups. Node structures can have several group memberships and be efficiently evaluated.
- **Finding coherent attribute subspaces:** We think that some information is hidden in attribute subspaces that can be obtained to express the meaning of the cluster. Overlapping is also allowed among attribute subspaces.
- **Without Redundancy:** Based on the Minimum Description Length (MDL) principle [11], our approach balances quality and redundancy in both graph structure and attribute space. Specifically, a node is only assigned to several graph clusters and an attribute is only part of multiple subspaces if this pays-off in terms of data compression. Thereby, only the truly relevant overlap useful to compress the data is reported as a result of our algorithm.
- **Automation:** Our information theoretic based algorithm relieves the user from the task of finding parameters to run the method on every specific data. The non-redundant overlapping clusters and coherent attribute subspace can be detected automatically.

The remainder of this paper is organized as follows: In the following section, we elaborate our coding scheme, which

is necessary to avoid parametrization and balances among quality and redundancy. Section 3 describes the algorithm in detail. Section 4 shows experiments and results. Related work is discussed in Section 5. The conclusion follows in Section 6.

## 2. COMPRESSING AN ATTRIBUTED GRAPH

To understand our coding scheme it is foremost important to outline what needs to be compressed in an attributed graph. From start, attributed graphs are an extension from general graphs by involving attributed information to each vertex. Therefore, two types of matrices are needed to model both structural connections in the graph and the attribute space of each node. Same as the general graph, the structure is mapped into an adjacency matrix, while the attributes of each vertex can be modelled as a matrix with rows denoting vertices and columns representing attributes. We call that matrix an *attribute matrix*. So far, an attributed graph is represented by an adjacency matrix and an attributed matrix that need to be compressed for efficiency and automatization. For simplicity in this paper, we focus on undirected unweighed graphs with categorical attributes, which means both matrices are symmetric.

### 2.1 Notations

Before we start with the coding scheme this section describes the used notation in the paper for clarification.

**DEFINITION 1 (ATTRIBUTED GRAPH).** *An attributed graph is defined as  $G = (V, E, \Lambda)$ , where  $V = \{v_1, v_2, \dots, v_N\}$  contains  $N$  vertices,  $E = \{(v_i, v_j), 1 \leq i \leq N, 1 \leq j \leq N, i \neq j\}$  are edges,  $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_T\}$  are  $T$  attributes of each vertex.  $\Theta = \{\theta_1, \theta_2, \dots, \theta_T\}$  represents domains of  $\Lambda$ .  $A$  is the adjacency matrix with  $a_{ij} = 1$  if  $(v_i, v_j) \in E$  and  $F$  is the attribute matrix with  $f_{ik}$  denotes the categorical value of the  $i$ th vertex in the  $k$ th attribute. An attribute graph is represented as  $G = (A, F)$  as well.*

In this paper, we aim at mining knowledge from the attributed graph by detecting non-redundant overlapping clusters. As attributed graphs possess both structural and attribute information, the cluster of such type of data covers both informations as well, which is defined in Definition 2. A cluster  $C$  is a subset of an attributed graph  $G$ . Specifically, the cluster not only needs to be densely connected but also contains a subgroup of attributes to describe the meaning of the cluster.

**DEFINITION 2 (CLUSTER IN AN ATTRIBUTED GRAPH).** *A cluster is defined as  $C = (V', E', \Lambda')$ , which is a subset of the attributed graph  $G$ , where  $V' \subseteq V$ ,  $E' \subseteq E$ ,  $\Lambda' \subseteq \Lambda$ .  $V' = \{v'_1, v'_2, \dots, v'_{N'}\}$  is a subset of  $V$ , which includes  $N'$  densely connected vertices and  $\Lambda' = \{\lambda'_1, \lambda'_2, \dots, \lambda'_S\}$  is a subset of  $\Lambda$ , which contains  $S$  attributes with coherent categories, where  $S \leq T$ .  $A_C \subseteq A$  is the subset adjacency matrix of  $C$  that only contains the vertices in  $C$  and  $F_C \subseteq F$  is the subset attribute matrix of  $C$ . Similarly, a Cluster is represented as  $C = (A_C, F_C)$ .*

Besides the own structure of a cluster, there are some edges connecting these clusters that are not included inside the clustering. Similarly to the edge structure, many attributes from the full-dimensional subspace are not assigned to any cluster. We define these areas as the non-cluster area

of an attributed graph in Definition 3, which consists of the elements lying outside any cluster.

**DEFINITION 3 (NON-CLUSTER AREA).** *The non-cluster area of an attributed graph  $G$  modelled by  $K$  clusters  $\{C_1, C_2, \dots, C_K\}$  is defined as  $U = U_A \cup U_F$ , where  $U_A$  is the Non-Cluster area in the adjacency matrix  $A$  and  $U_F$  is the non-cluster area in the attributed matrix  $F$ .  $U_A = A \setminus A'$ , where  $A' = A_{C_1} \cup A_{C_2} \dots \cup A_{C_K}$  is the combination of all structural elements appearing in  $\{C_1, C_2, \dots, C_K\}$  and  $U_F = U_{\lambda_1} \cup U_{\lambda_2} \dots \cup U_{\lambda_T}$ , where  $U_{\lambda_t}$  are entries of  $F$  in attribute  $\lambda_t$  which are not included in  $\{C_1, C_2, \dots, C_K\}$ .*

## 2.2 Coding Scheme

### 2.2.1 Information Theory Basics

**Entropy:** Entropy[8] is adopted to quantify the uncertainty of a given data set, which is defined by Eq. (1). In this equation,  $D$  represents the given data which contains  $n$  components with the probabilities denoted as  $\{p_1, \dots, p_n\}$ . High entropy indicates that the data is unpredictable because of the balanced distribution of each component and data dominated by some certain components provide low entropy, having a high level of predictability. In an attributed graph, we use entropy to measure the density of a graph structure and the compactness of the subspace attributes.

$$H(D) = - \sum_{i=1}^n p_i \cdot \log_2 p_i \quad (1)$$

**Minimum Description Length Principle:** As a lossless compression, Minimum Description Length [11] follows the assumption that the less coding length we adopt to describe the data, the more knowledge we can gain from it. Formally, the quality of a model can be identified from Eq.(2), where  $L(M)$  denotes the coding length for describing model  $M$  and its parameters, while  $L(D | M)$  represents the cost of coding the data  $D$  under model  $M$ .

$$L(M, D) = L(D | M) + L(M) \quad (2)$$

In the following, we elaborate the model and the data description costs necessary to compress an attributed graph in detail.

### 2.2.2 Data Description Cost $L(D | M)$

Suppose  $K$  clusters  $\{C_1, C_2, \dots, C_K\}$  are discovered from an attributed graph  $G$ . Under the clustering model, the attributed graph can be described as  $K$  clusters  $\{C_1, C_2, \dots, C_K\}$  and a non-cluster area  $U$ . Therefore the data description cost  $L(D | M)$  is equivalent to all costs of all clusters plus the non cluster area, as shown in Eq.(3).

$$L(D | M) = \sum_{i=1}^k CC(C_i) + CC(U) \quad (3)$$

**Coding cost of a cluster  $CC(C_i)$ :** Compressing an attributed graph is equivalent to compressing its adjacency matrix  $A$  and its attribute matrix  $F$ . A cluster  $C_i$  is represented by the subset adjacency matrix  $A_{C_i}$  and the subset attribute matrix  $F_{C_i}$ , where  $A_{C_i} \subset A$  and  $F_{C_i} \subset F$ . Then the coding cost of the cluster  $C_i$  is the sum of the structural coding cost  $CC^A(C_i)$  and the attribute coding cost  $CC^F(C_i)$ .

Firstly, cluster  $C_i$  is composed of densely connected vertices which equals to high probability 1s in subset adjacency matrix  $A_{C_i}$ . The average coding cost of the entries in matrix  $A_{C_i}$  is lower bounded by its entropy. Because we consider  $G$  an undirected graph, we only need to encode the entries of the upper triangular matrix. The coding cost of the structural information of the cluster  $C_i$  is described in Eq.4, where  $p_1(C_i)$  and  $p_0(C_i)$  stand for the probability of 1s and 0s in the subset adjacency matrix  $A_{C_i}$  respectively. And  $n_{C_i} = \frac{N_{C_i} \cdot (N_{C_i} - 1)}{2}$  refers to the number of entries in upper triangular matrix of  $A_{C_i}$ .

$$CC^A(C_i) = -n_{C_i} \cdot (p_1(C_i) \log_2 p_1(C_i) + p_0(C_i) \cdot \log_2 p_0(C_i)) \quad (4)$$

Secondly, the subspace of the densely connected vertices is described as a subset attribute matrix  $F_{C_i}$ . Originally, each vertex possesses a category in every attribute. In order to find the meaning of the cluster,  $S$  attributes which contain consolidated categories in each attribute are chosen from  $T$  attributes. In attribute matrix  $F_{C_i}$ , each vertex possesses a category in each attribute of the subspace. Equally, in cluster  $C_i$ , the attributes of a vertex can be represented as a category string with a size equals to the subspace. Figure 2 depicts a codebook which is adopted to encode the attribute information of cluster  $C_i$ . The codebook shows  $R$  groups of category strings of cluster  $C_i$  and the probabilities of each group are  $p_{g1}, p_{g2}, \dots, p_{gR}$ . Additionally, we use  $\log_2 n_{\theta_i}$  bits to encode each category string, and  $n_{\theta_i}$  is the number of categories in attribute  $i$  that included in subspace  $S$ . Then the coding cost of the attribute information of cluster  $C_i$  can be calculated from Eq.5. Finally, the coding cost of describing clusters is the sum of the coding cost of  $K$  clusters.

$$CC^F(C_i) = -R \cdot \sum_{i=1}^R p_{gi} \cdot \log_2 p_{gi} + R \cdot \sum_{i=1}^S \log_2 n_{\theta_i} \quad (5)$$

Subspace $\Lambda'$					
Group 1 :	a	a	...	a	$p_{g1}$
Group 2 :	a	a	...	b	$p_{g2}$
$\vdots$			...		...
Group R :	a	b	...	c	$p_{gR}$

Figure 2: The codebook of an attribute matrix

**Coding cost of the non-cluster area  $CC(U)$ :** The Non-cluster area consists of the elements in the adjacency matrix  $A$  and the attributed matrix  $F$  that are not contained in  $K$  clusters, which are represented as  $U_A$  and  $U_F$  respectively. Similarly, the coding cost of the non-cluster area  $CC(U)$  is the sum of the structural coding cost  $CC(U_A)$  and the attributed coding cost  $CC(U_F)$ , as shown in Eq.(6).

$$CC(NC) = CC(U_A) + CC(U_F) \quad (6)$$

We consider all elements of the structural non-cluster area  $U_A$  entirely and code them with a fixed coding order. The coding cost of the non-cluster area of the structural aspect  $CC(U_A)$  can be encoded as shown Eq.7. Here,  $p_1(U_A)$  is

the number of edges in  $U_A$  and  $p_0(U_A)$  is the number of no-edges in  $U_A$ . Due to the symmetry property of the matrix,  $n_{U_A} = \frac{N_{U_A} \cdot (N_{U_A} - 1)}{2}$  is equals to half of the elements in the non-cluster area.

$$CC(U_A) = -n_{U_A} \cdot (p_1(U_A) \cdot \log_2 p_1(U_A) + p_0(U_A) \cdot \log_2 p_0(U_A)). \quad (7)$$

In the non-cluster area of attribute aspect  $U_F$ , we encode the remaining categories of each attribute one by one. The coding cost  $CC(U_F)$  is calculated in Eq.8, where  $N_{\lambda_i}$  is the number of categories of attribute  $\lambda_i$  that are not assigned to any clusters, and  $p_{\theta_j}$  is the probability of an category  $\theta_j$  in the remaining elements of the corresponding attribute.

$$CC(U_F) = - \sum_{i=1}^T \sum_{j=1}^{\theta_j} N_{\lambda_i} \cdot p_{\theta_j} \cdot \log_2 p_{\theta_j}. \quad (8)$$

### 2.2.3 Model Cost $L(M)$

For encoding the model cost  $L(M)$  of the attributed graph, every cluster will be compressed in three aspects: the assignments of each vertex, the assignments of each attribute and the parameters of the clusters.

**Coding cost of vertices assignment  $CC_{IDV}(C_i)$ :** As overlapping is allowed in our proposed algorithm, a vertex can be assigned to multiple clusters. For each cluster, we adopt an assignment list to label the existence of vertices. An example for this is shown in Figure 3. When the vertex belongs to the cluster, the corresponding value in the list is set to “1”, otherwise set to “0”. Therefore, the coding cost of the vertices assignment for a cluster  $CC_{IDV}(C_i)$  is lower bounded by its entropy as shown in Eq.(9), where  $p_1(L)$  and  $p_0(L)$  denote the probability of 1 and 0 in the assignment list of cluster  $C_i$  respectively, and  $N$  is the number of vertices in the graph. Then the coding cost of the vertex assignments of the whole graph is the sum of cost for all clusters.

$$CC_{IDV}(C_i) = -N \cdot (p_1(L) \cdot \log_2 p_1(L) + p_0(L) \cdot \log_2 p_0(L)). \quad (9)$$

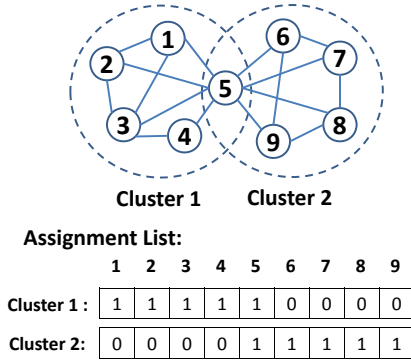


Figure 3: The assignment list of vertices

**Coding cost of attributes assignments  $CC_{IDF}(C_i)$ :** In our proposed algorithm, the corresponding attribute subspaces of clusters are also detected. Similarly, there is overlapping among the subspaces as well, which means an attribute can be grouped into multiple clusters. Here we also adopt an assignment list with size  $T$  to represent attributes that are contained in a subspace of a cluster, which is shown

in Eq.(10).

$$CC_{IDF}(C_i) = -T \cdot (p_1(L) \cdot \log_2 p_1(L) + p_0(L) \cdot \log_2 p_0(L)). \quad (10)$$

**Coding cost of parameters:** To ensure that the receiver acquires the complete information, all parameters need to be encoded to keep the lossless compression. The cost of these parameters can be calculated from Eq.(11), where  $n_p$  is the number of parameter and  $n_E$  is the number of entries.

$$CC_{para} = 0.5 \cdot n_p \cdot \log_2 n_E. \quad (11)$$

First, we need to encode the probability of “1”s and “0”s in each cluster  $C_i$  and of the non-cluster area  $U$ . Here  $n_p = K + 1$  and  $n_E$  is equals to half of the entries of the corresponding subset adjacency matrix  $A_{C_i}$ . Secondly, we encode the probabilities of the  $R$  groups in the codebook. Here  $n_p = R - 1$  and  $n_E$  are equals to the number of categories of the codebook. Thirdly, in the non-cluster area of the attribute matrix, we encode the probabilities of all categories in each attribute as parameter, for each attribute  $i$ . For this  $n_p = \theta_j - 1$  and  $n_E$  are equals to the size of the remaining categories of attribute  $i$ . Finally, we encode the probabilities of the assignment lists as parameters as well, here  $n_p = 1$  and  $n_E$  are equals to the size of the list.

## 3. ALGORITHM IROC

In this section, we propose an effective and efficient searching algorithm IROC to detect overlapping clusters in an attributed graph. The initialization phase and the refinement phase are the two key steps in IROC. The initialization phase is again divided into two subroutines for a) creating graph substructures and b) finding their coherent attribute subspace. The refinement phase takes the responsibility of improving the quality of initial clusters by a) removing redundant parts of the cluster and b) reassigning vertices between two clusters.

### 3.1 Initialization

**Creating Graph Substructures:** The idea for the implicit creation of an overlapping graph structure is to create small redundant subgraphs. Each small subgraphs is simply composed of a vertex and all its neighbors. This is an intuitive way to obtain all these small redundant subgraphs. Therefore,  $N$  subgraphs with overlapping vertices are extracted from an attributed graph  $G$ , where  $N$  is the number of vertices in  $G$ . First of all, in order to eliminate over much redundancies we select  $K_s$  subgraphs as initial rough clusters based on the MDL principle automatically. The procedure is shown in Algorithm 1. To be specific, we consider  $N$  subgraphs as rough clusters. And we consider there is no clusters in the attributed graph  $G$ . Then we add a rough cluster to the clusters  $C$  and calculate the coding cost by Eq.(2) one by one. After that we select the cluster with the minimum coding cost as the first cluster. We keep the selected cluster in the clusters  $C$ . Analogously, in the remaining  $N - 1$  candidates, we try to add each candidate as the second cluster and choose the one with the minimum coding cost repetitively. This process stops until all vertices are processed, so that  $K_s$  initial clusters are chosen automatically.

**Finding a Coherent Attribute Subspace:** After obtaining  $K_s$  rough clusters, we also need to find out the attribute

subspace of each rough clusters. Algorithm 2 demonstrates the progress of searching an attribute subspace of a cluster  $C_i$  based on the MDL principle. Every vertex of a rough cluster possesses a category in each attribute. For each cluster, we calculate the entropy of each attribute to measure the purity of the categories in the attribute. Entropy in this case refers to the consistence of categories of an attribute. The lower the value, the higher the consistency of the attribute. Thus we order the attributes in ascending order of their entropies. If some attributes possesses the same value, they are formed into groups and considered as one candidate. Thus the candidates of attributes are form as  $\Lambda^o = \{\Lambda_1^o, \Lambda_2^o, \dots, \Lambda_{T_g}^o\}$ . Based on the prerequisite that the subspace of the cluster is initialized with an empty set, we add the ordered attribute candidates to the subspace one by one, and meanwhile calculate the coding cost of the whole graph  $G$  by Eq.(2). Then we select the attribute subspace  $\Lambda'$  of a cluster  $C_i$  with the minimum coding cost. The attribute subspace of the cluster is fixed after chosen. In the same way, the attribute subspace of other  $K_s - 1$  clusters are chosen.

---

#### Algorithm 1 Creating Subgraphs

---

**Input:** Attributed Graph  $G$  with  $N$  Vertices and  $T$  Attributes  
**Output:**  $K_s$  Rough Clusters:  $C = \{C_1, C_2, \dots, C_{K_s}\}$   
1: Construct  $N$  subgraphs  $SS = \{ss_1, ss_2, \dots, ss_N\}$  as initial  $N$  clusters;  
2:  $C \leftarrow \emptyset$ ;  
3: **while** All vertices are ergodic **do**  
4:   **for** All substructures in  $SS$  **do**  
5:     Calculate coding cost of  $ss_i \cup C$ ;  
6:   **end for**  
7:   Select  $ss_m$  with minimum coding cost,  $ss_m \cup C$ ;  
8:   Remove  $ss_m$  from  $SS$ ;  
9: **end while**  
10: **return**  $K_s$  Rough Clusters:  $C = \{C_1, C_2, \dots, C_{K_s}\}$ .

---



---

#### Algorithm 2 Finding Subspace

---

**Input:** A Cluster  $C_i$  with  $N_c$  Vertices and  $T$  Attributes  
**Output:** Subspace  $\Lambda' = \{\lambda'_1, \lambda'_2, \dots, \lambda'_S\}$   
1:  $\Lambda' \leftarrow \emptyset$ ;  
2: **for**  $i$  from 1 to  $T$  **do**  
3:   Calculate entropy of each attribute  $\lambda_i$ ;  
4: **end for**  
5: Group attributes with same entropy and arrange attributes in ascending order  $\Lambda^o = \{\Lambda_1^o, \Lambda_2^o, \dots, \Lambda_{T_g}^o\}$ ;  
6: **for**  $i$  from 1 to  $T_g$  **do**  
7:    $\Lambda_i^o \cup \Lambda'$ , and calculate coding cost of all clusters  $CC$ .  
8:   **if**  $CC$  increases **then**  
9:     break;  
10:   **end if**  
11: **end for**  
12: **return** Subspace  $\Lambda' = \{\lambda'_1, \lambda'_2, \dots, \lambda'_S\}$ .

---

## 3.2 Refinement

**Removing Redundancy:** After the initialization step, we receive  $K_s$  small attributed clusters which contain *redundant* information. These redundancies serve to generate and find the overlapping in the beginning but are later unwelcome.

Therefore our heuristic bottom up algorithm needs to merge these small initial rough clusters to remove redundancies. We define an information-theoretic based similarity which measures the degree of the redundancy between every pair of clusters. Due to the fact that both structural and attribute information are present, we measure the similarity of two clusters as shown in Eq. (12):

$$Sim(C_i, C_j) = H(O_A) + H(O_F). \quad (12)$$

where  $H(\cdot)$  is entropy defined in Eq. (1),  $O_A$  and  $O_F$  are the structural and attribute overlapping part of cluster  $C_i$  and  $C_j$  respectively. This quality function tackles the chief problem of merging two cluster: the amount of overlaps in structural and attribute information. How can two cluster be merged without creating new redundancy if they are overlapping in structure and attribute space? On the structural aspect, entropy payoff the denser the overlapping area. Also the smaller the entropy, the denser the overlapping part of the cluster. Thus the smaller entropy of structural overlapping part leads us to detect dense clusters. While on the attribute side, the entropy measures the purity of categories in the overlapping area of the attributes. This ensures that the cluster is provided with same categories in the attributes. The smaller the entropy is, the higher the similarity of the attributes in the overlapping part is. Thus the smaller entropy of attribute overlapping part leads us to detect clusters with coherent meaning. Therefore, the defined similarity balances the entropy of the two aspects, which guides us to merge the two most similar clusters. And in every search run we merge the pair of clusters that have a minimal similarity.

**Assigning Vertices:** Obviously, merging two clusters  $C_i$  and  $C_j$  to form a new cluster  $C_{new}$  is able to remove redundancy. However, such merging may not eliminate all redundancies. Depending on how accurate the removing of the redundancy works, we need to modify the cluster further in a second refinement step which is shown in Algorithm 3. For all vertices  $\{v_1, v_2, \dots, v_{N_{C_{new}}}\}$  in  $C_{new}$ , we try to split a vertex  $v_1$  from  $C_{new}$  and consider it as a new cluster  $C_{split} = \{v_1\}$ . Then we find the subspace of  $C_{split}$  and refine the subspace of  $C_{new} = \{v_2, \dots, v_{N_{C_{new}}}\}$ . If such process reduce the coding cost, we keep the modification and try to move the next vertex  $v_2$  from  $C_{new}$  to  $C_{split}$ . If the coding cost is not reduced, we move the vertex  $v_1$  back to  $C_{new}$ . The refinement of cluster  $C_{new}$  ends when the coding cost achieves its local minimum. If cluster  $C_{split}$  is not empty, we treat  $C_{split}$  as a new candidate and add it to the clusters.

## 3.3 IROC

The overall procedure of IROC is shown in Algorithm 4. First, we automatically select  $K_s$  rough clusters and search their attribute subspace as described in the initialization phase. Then we calculate the similarity of every pair of clusters, and merge the two cluster with a minimum similarity. After that a new cluster  $C_{new}$  is formed and we find the subspace of it. Then we try to assign vertices from  $C_{new}$  to  $C_{split}$  under the control of MDL. And we consider  $C_{split}$  as a new cluster and recalculate the similarity of every pair of clusters. The merging process continues iteratively and is ended when the coding cost of all clusters achieve its local minimum. Finally,  $K$  clusters with coherent attribute subspaces without redundancy are output.

---

**Algorithm 3 Assigning Vertices**

---

**Input:** Cluster  $C_{new}$   
**Output:** Cluster  $C_{new}$  and Cluster  $C_{split}$

- 1: Create a cluster  $C_{split}$ ,  $C_{split} \leftarrow \emptyset$ ;
- 2: Calculate coding cost of all the clusters  $CC$ ;
- 3: **for** All nodes in  $C_{new}$  **do**
- 4:    $v \in C_{new}$ , remove  $v$  from  $C_{new}$ , add  $v$  to  $C_{split}$ ;
- 5:   **Finding Subspace** of  $C_{split}$  and  $C_{new}$ ;
- 6:   Calculate coding cost of all the clusters  $CC_{new}$
- 7:   **if**  $CC_{new} > CC$  **then**
- 8:     Restore  $v$  to  $C_{new}$ ;
- 9:   **end if**
- 10: **end for**
- 11: **return**  $C_{new}$  and  $C_{split}$ .

---



---

**Algorithm 4 IROC**

---

**Input:** Attributed Graph  $G$   
**Output:**  $K$  Clusters with Subspace  $C = C_1, C_2, \dots, C_K$

- 1: **Creating Subgraphs**  $C = C_1, C_2, \dots, C_{K_s}$ ;
- 2: **for**  $i$  from 1 to  $K_s$  **do**
- 3:   **Finding Subspace** of  $C_i$
- 4: **end for**
- 5: **while** Converge **do**
- 6:   Calculate similarity of every pair of clusters;
- 7:   Merge two most similar clusters as  $C_{new}$ ;
- 8:   **Finding Subspace** of  $C_{new}$ ;
- 9:   **Assigning Vertices** of  $C_{new}$ ;
- 10: **end while**
- 11: **return**  $C = C_1, C_2, \dots, C_K$ .

---

### 3.4 Complexity Analysis

Suppose we have an attributed graph  $G$  with  $N$  vertices,  $E$  edges and  $T$  attributes for each vertex. We first analyse the complexity of the coding when we have  $K$  clusters. For each clusters we need to count the edges of the cluster and go through its vertices and attributes to get the probability distributions. The complexity of these processes for all clusters are  $O((K+1) \cdot (A_{vE} + A_{vN} \cdot A_{vT}))$ , where  $A_{vE} < E$  is the average number of edges,  $A_{vN} < N$  is the average number of vertices and  $A_{vT} < T$  is the average number of subspace attributes. The multiplier factor is  $K + 1$  considering the no-cluster area. Since the number of cluster is  $K \ll N$ , the complexity of the proposed coding scheme is  $O(E + N \cdot T)$ .

In initialization phase of IROC, we greedily choose the initial clusters, since their attributes are not considered in this part, the complexities are  $O(K_s \cdot N)$  and  $O(N \cdot E)$  respectively, where  $K_s$  is the number of initial clusters. Then we need to find the subspace for each cluster. In this part only attributes are considered, its complexity is  $O(K_s \cdot N \cdot T^2)$ . The reason is that for each cluster we need to calculate the coding cost  $T$  times. Finally, the complexity of the full initialization phase (both parts combined) is  $O(K_s(N + N \cdot T^2))$  for the random one and  $O(K_s \cdot N(E + T^2))$  for the greedy approach.

In the full refinement step, we need to process  $K_s^2$  pairs of clusters. In each merging and splitting process, we need to move vertices to other clusters and then calculate the subspace of the new clusters. Therefore, we need  $O(N^2 \cdot T^2)$  time to do so. Finally, the complexity in this step is  $K_s^2 \cdot O(N^2 \cdot T^2)$ .

## 4. EXPERIMENTS

In this section, we evaluate our proposed algorithm IROC on both synthetic data sets and real data sets and compare it with 4 relevant algorithms. The code of the comparison methods are obtained from the authors. Experiments have been performed on a workstation with 2.9GHz Intel Core i7 and 8G memory.

### 4.1 Synthetic Data sets

#### 4.1.1 Data sets

The sketches of three synthetic data sets are shown in Tab. 1. Each circle represents a densely connected cluster and the shadow part denotes the overlapping part in which vertices are assigned to multiple clusters. The synthetic attributed graphs are generated with the following parameters: Each cluster contains 200 vertices ( $n_c = 200$ );  $n_o$  is the number of vertices in the overlapping part; the density of edges in and between clusters are 0.8 and 0.1 respectively ( $d_c = 0.8, d_b = 0.1$ ). The subspace of attributes of the three synthetic data sets are also shown in Table 1. The number of dimensionality of the subspace of each cluster is denoted as  $n_d$  and 2 dimensions are overlapping ( $n_{od} = 2$ ). Each cluster contains nearly same category in its subspace attributes and  $r = 0.05$  of them are randomly generated with other categories. The categories which do not belong to the subspace are randomly generated.

#### 4.1.2 Evaluation of Clustering

In this section, we evaluate IROC on three synthetic data sets and compare it to four relevant algorithms. PICS [1] is a compression based algorithm which clusters both vertices and binary attributes. BAGC [16] is a probability model based attributed graph partition method. DB-CSC [7] is a density based algorithm which aims at detecting dense clusters with coherent subspace of attributes. It is designed for graph with numerical attributes, which chooses the attribute neighborhood if the distance of attributes of two vertices is smaller than a threshold  $\epsilon$ . The distance is defined as the maximum difference of all the attributes like  $dist(x, y) = \max_{i \in \{1, \dots, T\}} |x[i] - y[i]|$ . We integerize the categorical attributes of our synthetic data for DB-CSC and set  $\epsilon < 1$ , which means the attribute neighbors with exactly same category will be chosen. CONGO is an overlapping community detection method without considering node attributes which is adopted to compare the ability of detecting overlapping communities in structural part. Due to the multiple clustering assignments, we adopt F1-Measure to evaluate these algorithms on clustering vertices. F1-Measure is computed as the harmonic mean of Precision and Recall. Precision measures the accuracy of the detected clusters and Recall measures whether all clusters are detected.

From Tab. 2, it is obvious that our proposed algorithm IROC outperforms the other algorithms. Generally, PICS and BAGC are not able to detect overlapping parts of graphs. DB-CSC outputs many small clusters with several overlapping vertices. However 6 parameters have to be set. CONGO performs worse than IROC as well, which provides redundant results. Besides, it needs the number of clusters and the depth of local betweenness. Specifically, in *Syn1* IROC achieves perfect 2 clusters without any parameters, as shown in Fig. 4a. From Fig. 4b, parameter-free algorithm PICS outputs 8 clusters, which splits two big cluster to several



Table 1: Parameter Settings for Generating Synthetic Data Set

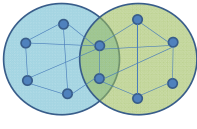
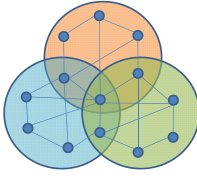
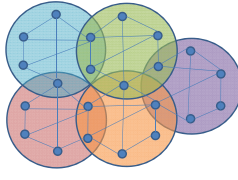
Datasets	Syn1	Syn2	Syn3
Sketch			
Structure Setting	$n_c = 200; n_o = 10;$ $d_c = 0.8; d_b = 0.1$	$n_c = 200; n_o = 10;$ $d_c = 0.8; d_b = 0.1$	$n_c = 200; n_o = 10;$ $d_c = 0.8; d_b = 0.1$
Attribute Setting	$n_d = 5; n_{od} = 2; r = 0.05$	$n_d = 8; n_{od} = 2; r = 0.05$	$n_d = 10; n_{od} = 2; r = 0.05$
Subspace Setting	$C1 : \{1, 2, \dots, 5\}$ $C2 : \{3, 4, \dots, 8\}$	$C1 : \{1, 2, \dots, 8\}$ $C2 : \{7, 8, \dots, 14\}$ $C3 : \{13, 14, \dots, 20\}$	$C1 : \{1, 2, \dots, 10\}$ $C2 : \{9, 10, \dots, 18\}$ $C3 : \{17, 18, \dots, 26\}$ $C4 : \{25, 26, \dots, 34\}$ $C5 : \{33, 34, \dots, 42\}$

Table 2: Evaluation Overlapping Clusters of Synthetic Data Sets

Algorithms	Syn1			Syn2			Syn3		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
IROC	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0.973</b>	<b>0.986</b>	<b>1</b>	<b>0.963</b>	<b>0.981</b>
PICS	1	0.322	0.487	1	0.732	0.846	0.563	0.670	0.612
DB-CSC	0.895	0.826	0.859	—	—	—	—	—	—
BAGC	1	0.947	0.973	0.955	0.607	0.742	0.490	0.722	0.584
CONGO	0.524	1	0.688	0.392	1	0.563	—	—	—

small ones. BAGC achieves 2 clusters, which cannot detect overlapping part. DB-CSC outputs 19 clusters and some clusters contain less than ten vertices. We run DB-CSC with  $\epsilon = 0.5$ ,  $k_{min} = 4$ ,  $min_{pts} = 5$ ,  $r_{obj} = 0.1$ ,  $r_{dim} = 0.1$  and  $s_{min} = 1$ . For CONGO, we set the number of clusters to 2 and the depth of local betweenness  $h = 2$ . It detects a very large overlap between two clusters, which is far away from the truth. For *Syn2*, overlap exists among all the three clusters. The number of dimensions of each cluster is increased to 8. IROC achieves 3 perfect clusters. It also successfully detects the vertices which are assigned to all three clusters. PICS outputs 6 clusters of vertices without overlapping. BAGC produces 3 cluster. DB-CSC achieves no result after adjusting 6 parameters several times and running the algorithm several days. The reason might be that DB-CSC is not applicable for dense graph with higher dimensional attributes. CONGO outputs 3 clusters with very large overlapping parameterized with  $h = 2$ . *Syn3* is more complex than the two synthetic data sets before. Similarly, IROC output 5 good clusters. PICS partitions vertices to 7 clusters. BAGC partitions vertices to 5 clusters. DB-CSC still cannot output any results after increasing the number of dimension of each cluster to 10. Due to the increased size of the data, CONGO can not achieve any results after several days running.

#### 4.1.3 Evaluation of Subspace

CONGO is designed for general graphs without node attributes and BAGC is not able to detect subspaces of attributes. PICS is an algorithm which aims at clustering the binary attributes and not at finding attributed subspaces of certain clusters. For example, Fig.4 shows the attributed matrix of *Syn1* after running IROC and PICS. Specifically,

the size of the attributed matrix of IROC is  $380 \times 8$ , where black points mean that the attribute is included in a subspace of a cluster. Two black blocks represent the clustering results of our IROC, which clearly shows the subspace of attributes and both overlaps on vertices and attributes. While the size of the attribute matrix of IROC is  $380 \times 35$ . Black points mean that the vertices have corresponding categories and red lines demonstrate there are 5 attributes clusters. It is difficult to judge which attributed clusters are part of subspace to represent the meaning of the cluster. In addition, subspaces of various clusters may contain overlap, while the clustering of attributes is not able to achieve overlaps. Besides, it is hard to obtain the binary image of DB-CSC due to the fact that too many clusters with subspace are produced. Therefore, we compare IROC with DB-CSC by evaluating F1-Measure of subspaces. Obviously, IROC achieves perfect results on every synthetic data set. The F1-Measure of DB-CSC on *Syn1* is 0.13, as there are many small clusters and the clustering results on structural part does not lead to detect accurate subspace. Besides, PICS obtain 10 attributes clusters on *Syn2* and 9 attributes clusters on *Syn3*. Similarly, they provide no information of which attributed clusters are part of subspace to represent the meaning of the cluster.

## 4.2 Real Data sets

### 4.2.1 Facebook Data

Facebook data sets are obtained from SNAP data sets [9]. Specifically, each Facebook data set is an ego-network of a selected node. For example, a network named “1912” is generated by a node with id 1912 and all its neighbors. Characters of vertices include birthday, education information, lan-

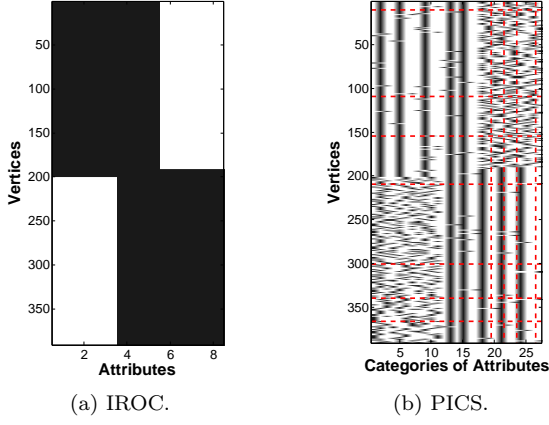


Figure 4: Clustering Results of Syn1 in Attributed Matrix.

Table 3: F1-Measure of Facebook Data Sets

Data sets	IROC	PICS	BAGC
<i>Ego – network</i> "1912"	<b>0.231</b>	0.224	0.229
<i>Ego – network</i> "107"	<b>0.141</b>	0.101	0.118

guage, hometown, work information .etc. Each kind of property contains several anonymity items. The original node attributes are in binary form, and every single item is considered as an attribute. The value "1" stands for the node that contains this item information and value "0" vise verse. In this paper, we consider each type of property as attribute and the items in each property are defined as categories. In this section, we test two data sets: Ego-network "1912" and Ego-network "107". For the same reason with *Syn3*, we are not able to get the results from DB-CSC and CONGO after running the experiments for several days. Each data set is separated into several "circles", but not all the vertices are included. That means only part of the vertices' label are available, thus we still use F1-Measure to compare the algorithms: IROC, PICS and BAGC.

There are 755 vertices in ego-network "1912", and each vertex has 22 attributes. Each attribute contains several categories, for example, there are 22 categories in attribute "birthday" and 19 categories in "education classes". The data set has already been labeled as 46 circles. Obviously, Tab. 3 demonstrates that our proposed method achieves the best F1-Measure value by generating 12 clusters. PICS generates 8 clusters. The result of BAGC is acquired by setting the number of clusters to 46, which equals to the given number of clusters. Ego-network "107" contains 1045 vertices, 23 attributes and 9 circles. Tab. 3 demonstrates that our proposed method IROC achieves the best F1-Measure value by generating 17 clusters. PICS partitions vertices into 15 clusters, and the result of BAGC is acquired by setting the number of clusters as 9. Therefore, Tab. 3 demonstrates that IROC has a big advantage in detecting vertex clusters.

In addition, IROC discovers overlapping vertices between pairs of clusters. Take the Ego-network "1912" data as an example, the analysis of overlapping is shown in Tab. 4. Specifically, Fig. 5 shows a diagram of the overlapping part between cluster  $C_2$  and cluster  $C_{11}$ . There are four overlapping vertices "1941", "2347", "2468" and "2543" forming a clique. Other vertices in  $C_2$  and  $C_{11}$  are both closely con-

nected to the clique. Therefore, the two clusters  $C_2$  and  $C_{11}$  share the information of overlapping part and it is reasonable to assign overlapping vertices to both clusters.

Table 4: Overlapping of Ego-network "1912"

Cluster ID	Overlapping Clusters ID(No.Vertices)
$C_1$	$C_5(1)$
$C_2$	$C_3(1), C_4(4), C_5(23), C_6(11), C_7(1), C_8(3), C_9(2), C_{10}(6), C_{11}(4), C_{12}(9)$
$C_3$	$C_6(1), C_{11}(15), C_{12}(4)$
$C_4$	$C_5(1), C_6(1), C_{11}(11), C_{12}(12)$
$C_5$	$C_6(1), C_7(1), C_9(1), C_{12}(3)$
$C_6$	$C_{10}(1), C_{12}(3)$
$C_{10}$	$C_{11}(1)$

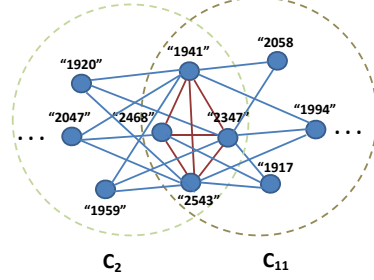


Figure 5: Overlapping Between Cluster 2 and Cluster 11 of Ego-network "1912".

Betweenness centrality of a vertex is the ratio of the number of the shortest paths from  $v_m$  to  $v_n$  through vertex  $v$  to the number of all shortest paths from  $v_m$  to  $v_n$  which is shown in Eq. (13).

$$Betweenness(v) = \sum_{v_m \neq v \neq v_n} \frac{Path_{v_m v_n}(v)}{Path_{v_m v_n}}. \quad (13)$$

Where betweenness centrality measures the ability of a human communicating with other humans in a social network [3]. That is to say, vertex with high betweenness plays the role of bridge. We calculate betweenness centrality of all the vertices, and rank them in descending order. Take the overlapping part of  $C_2$  and  $C_{11}$  as an example, the rank of the betweenness values of "1941", "2347", "2468" and "2543" is 5, 4, 20 and 3 separately, which possesses really high betweenness centrality values. Then these four persons are very outgoing and sociable. They communicate with various communities, so that they are assigned to multiple clusters. Besides, the betweenness values of all overlapping vertices are large, which implies that these vertices possess higher ability to communicate with other vertices thus assigning into multiple clusters.

Regarding the attributes, BAGC is not able to detect any subspace of clusters. PICS detects 8 attributes clusters in Ego-network "1912" and 11 attributes clusters in Ego-network "107". Our IROC detects a subspace of attributes for each cluster. Tab. 5 shows the attributes subspaces of Ego-network "1912". The table shows that overlaps also exist between subspaces. For example,  $C_5$  and  $C_6$  have one vertex overlapping, and meanwhile their subspaces also overlap on attribute "middle name".



Table 5: Subspace detected by IROC of Ego-network "1912"

Cluster ID	Subspace
$C_1$	all 22 attributes
$C_2$	"middle name"
$C_3$	"middle name", "work projects"
$C_4$	all 22 attributes
$C_5$	"middle name", "work projects"
$C_6$	"middle name"
$C_7$	all 22 attributes
$C_8$	all 22 attributes
$C_9$	all 22 attributes
$C_{10}$	"work projects"
$C_{11}$	"work projects"
$C_{12}$	"middle name"

Table 6: F1-Measure of Google+ Data Sets

Data sets	IROC	PICS	BAGC	DBCSC
gp7	<b>0.248</b>	0.205	0.236	0.106
gp53	<b>0.289</b>	0.125	0.130	0.074

#### 4.2.2 Google+ Data

Google+ data sets are obtained from SNAP data sets [9] as well. Specifically, each Google+ data set is also an ego-network of a selected node. Similarly, we transfer the given binary feature to categorical feature which contains 6 attributes: gender, institution, job title, last name, place and university. Similarly, part of labels are given as circles, thus we compare F1-Measure of the algorithms: IROC, PICS, BAGC and DBCSC.

Tab. 6 shows that IROC achieves the best clustering result among all the algorithms. Take the data set "gp53" which contains 1084 vertices and 6 attributes for example, our proposed algorithm IROC detects 17 clusters with overlapping and without redundancy. Each cluster is provided with an attribute subspace which represents the meaning of the clusters. For example, people in one cluster are connected densely and all of them are from the same university, or people in one cluster are densely connected and all of them are from the same institution. Moreover, PICS outputs 12 vertices clusters and 6 attributes clusters. We set the number of cluster of BAGC to 4, which equals to the given number of circles. DBCSC outputs 15 clusters parametrized with  $\epsilon = 0.5$ ,  $k_{min} = 2$ ,  $min_{pts} = 3$ ,  $r_{obj} = 0.3$ ,  $r_{dim} = 0.3$ ,  $s_{min} = 3$ . All the 15 detected clusters are provided with a subspace. However, there are three clusters with exactly same vertices which contain redundant information.

## 5. RELATED WORK AND DISCUSSION

Our proposed algorithm IROC is design for clustering the attributed graph under the permission of overlapping in both vertices and the subspace of attributes. The related work therefore comprises two parts: attributed graph clustering and overlapping community detection methods.

### 5.1 Attributed Graph Clustering

Attributed graph is an extension of general graph by involving the attributes of the vertices. The key point of mining such complex data is to combine structural connections and characteristics of the vertices. Paper [18] augments graphs by considering each attribute as vertex. A Random walk is utilized on the augmented graph so as to create a

unified similarity measure which combines structural and attribute information. However, the algorithm is carried out in a K-Medoids framework and partitions the attributed graph. It also needs to input the number of clusters and parameters for the random walk, such as number of steps and the probability to go back. Moreover, paper [19] proposes an incremental algorithm to improve efficiency of [18]. Paper [16] proposes an algorithm named BAGC based on Bayesian probabilistic model to partition attributed graphs. Structural and attributed information are fused by the probabilistic model and clustering problem is transferred to a probabilistic inference problem. Similarly, the algorithm needs to input the number of clusters, and many other parameters to construct the necessary probability distribution. Obviously, these partition based methods can not detect any overlapping of clusters. And they do not find coherent attributed subspace of cluster. For these partitioning approaches, we choose BAGC as a comparison method.

Guenemann et al. proposes an algorithm named DB-CSC (Density-Based Combined Subspace Clustering) [7] which is based on the classical density-based clustering algorithm DBSCAN [2]. The new proposed DB-CSC inherits the advantages of DBSCAN which can detect clusters of arbitrary shapes and sizes. It defines a combined local neighborhood by finding vertices, which belong to the intersects of  $k$ -neighborhood of vertices and  $\epsilon$ -neighborhood of subspace of the attributes. Based on the new defined neighborhood, some density related properties like high local density, local connected and maximality is defined, thus the fulfilled clusters can be detected. Instead of giving the number of clusters, DB-CSC needs parameters like  $\epsilon$ -neighborhood,  $k$ -neighborhood and a minimum number Minpts. In order to remove redundant clusters of the above two algorithms, the authors proposed a definition which is used to calculate redundancy between clusters. After adopting the strategy of removing redundancy clusters, the combined new algorithms need more parameters,  $robj$  and  $rdim$  which measure how much overlap between clusters is allowed without redundancy. Therefore, DB-CSC needs an isolate process and set several parameters to remove redundancy. If the parameters is set unpropertied, redundancy still exists. In comparison, IROC obtains the non-redundant results without setting any parameters. Moreover, as mentioned in experiments part, judging from the distance defined in the paper, DB-CSC is defined to deal with the attributed graphs with numerical attributes. But categorical attributes like gender, hobby .etc are common in social network data set. Therefore, IROC is able to detect overlapping clusters of the categorical attributed graph and meanwhile find the coherent attribute subspace of each cluster.

PICS[1] is also a parameter free algorithm based on MDL principle. It is able to mine cohesive clusters from an attributed graph with similar connectivity patterns and homogeneous attributes. However, it can not detect any overlapping and it cluster the vertices and attributes separately. Thus it is hard to find out which subspace belongs to which clusters. Additionally, Sun et al. [13] proposes a model-based method to clustering heterogeneous information networks which are containing incomplete attributes and multiple link relations. Also marginally related to our method are the approaches [12][10][15] achieving numerous small cohesive subgraphs, which aim to discover a correlation between node attributes and small subgraphs. Paper [14] summarizes

multi-relational attributed graphs by aggregating nodes by using selected attributes and relations.

## 5.2 Detecting Overlapping Communities

The key point of acquiring overlapping clusters is how to assign a vertex to multiple labels. In first instance, [4] reveals overlapping phenomena of complex networks in nature, and achieves overlapping communities by seeking  $k$ -cliques which contains overlapping vertices. Paper [17] proposes an algorithm based on matrix factorization. The assignment of each vertices is stored as probability in a matrix with a number of dimensions equal to the number of the community. By fuzzy allocation, overlap between communities is achieved. CONGA[5] proposed by Gregory is an algorithm which aims to detect overlapping communities by iteratively calculating two betweenness centrality based concepts “edge betweenness” and “split betweenness” of all edges and vertices respectively and removing the edge or splitting the vertex with the highest value until no edges remain. As betweenness centrality is a global measure of vertices in a graph, the calculation of the two concepts depends on counting the number of shortest paths of all pairs of vertices, which is really time consuming. In order to speed up the algorithm CONGA, the author proposes an algorithm named CONGO[6] by calculating local betweenness instead of global betweenness. In the new algorithm, a parameter  $h$  is added, which is a threshold that the shortest path which is more than  $h$  is ignored. Thus the concepts only need to recalculate locally to save time complexity. Both CONGA and CONGO need user to predetermined the number of clusters  $k$ . In this paper, we compare IROC with CONGO to prove the efficiency of our algorithm to detect overlapping clusters. Actually, the overlapping community detection algorithms which are mentioned above are all only considering the structural information of graph with no additional attributes.

## 6. CONCLUSION

Summarizing this work, we introduced the first solution that is explicitly able to find meaningful overlapping in the graph structure as well as in its respective attribute subspace. We outlined the importance of these overlapping communities especially for attributed graphs and the information gain that can be received by it. Our method IROC applied the concept of information theoretic measures like entropy and an Minimum Description length (MDL) formula designed for this challenge to elegantly avoid a) redundancy in the attribute space as well as in the network itself and b) the need to set in an unsupervised setting typically unknown input parameters to achieve good results. Our experiments clearly showed that IROC is able to outperform all relevant comparison methods on synthetic data and on real world data of social networks.

## 7. REFERENCES

- [1] L. Akoglu, H. Tong, B. Meeder, and C. Faloutsos. Pics: Parameter-free identification of cohesive subgroups in large attributed graphs. In *SDM*, pages 439–450, 2012.
- [2] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.
- [3] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, March 1977.
- [4] F. I. T. V. Gergely Palla, Derenyi Imre. Uncovering the overlapping community structure of complex networks in nature and society. *Nature Letter*, abs/1010.1523:814–818, 2005.
- [5] S. Gregory. An algorithm to find overlapping community structure in networks. In *PKDD*, pages 91–102, 2007.
- [6] S. Gregory. A fast algorithm to find overlapping communities in networks. In *ECML/PKDD (1)*, pages 408–423, 2008.
- [7] S. Günnemann, B. Boden, and T. Seidl. Db-csc: A density-based approach for subspace clustering in graphs with feature vectors. In *ECML/PKDD (1)*, pages 565–580, 2011.
- [8] S. Ihara. *Information theory for continuous systems*. World Scientific, 1993.
- [9] J. J. McAuley and J. Leskovec. Learning to discover social circles in ego networks. In *NIPS*, pages 548–556, 2012.
- [10] F. Moser, R. Colak, A. Rafiey, and M. Ester. Mining cohesive patterns from graphs with feature vectors. In *SDM*, pages 593–604, 2009.
- [11] J. Rissanen. *Information and Complexity in Statistical Modeling*. Springer, 2007.
- [12] A. Silva, W. M. Jr., and M. J. Zaki. Mining attribute-structure correlated patterns in large attributed graphs. *PVLDB*, 5(5):466–477, 2012.
- [13] Y. Sun, C. C. Aggarwal, and J. Han. Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. *PVLDB*, 5(5):394–405, 2012.
- [14] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *SIGMOD Conference*, pages 567–580, 2008.
- [15] H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In *KDD*, pages 737–746, 2007.
- [16] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. In *SIGMOD Conference*, pages 505–516, 2012.
- [17] Y. Zhang and D.-Y. Yeung. Overlapping community detection via bounded nonnegative matrix tri-factorization. In *KDD*, pages 606–614, 2012.
- [18] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.
- [19] Y. Zhou, H. Cheng, and J. X. Yu. Clustering large attributed graphs: An efficient incremental approach. In *ICDM*, pages 689–698, 2010.