Joint Cluster Analysis of Attribute and Relationship Data Without A-Priori Specification of the Number of Clusters

Flavia Moser, Rong Ge, Martin Ester
School of Computing Science
Simon Fraser University
fmoser, rge, ester @cs.sfu.ca

ABSTRACT

In many applications, attribute and relationship data are available, carrying complementary information about real world entities. In such cases, a joint analysis of both types of data can yield more accurate results than classical clustering algorithms that either use only attribute data or only relationship (graph) data. The Connected k-Center (CkC) has been proposed as the first joint cluster analysis model to discover k clusters which are cohesive on both attribute and relationship data. However, it is well-known that prior knowledge on the number of clusters is often unavailable in applications such as community identification and hotspot analysis. In this paper, we introduce and formalize the problem of discovering an a-priori unspecified number of clusters in the context of joint cluster analysis of attribute and relationship data, called Connected X Clusters (CXC) problem. True clusters are assumed to be compact and distinctive from their neighboring clusters in terms of attribute data and internally connected in terms of relationship data. Different from classical attribute-based clustering methods, the neighborhood of clusters is not defined in terms of attribute data but in terms of relationship data. To efficiently solve the CXC problem, we present JointClust, an algorithm which adopts a dynamic two-phase approach. In the first phase, we find so called *cluster atoms.* We provide a probability analysis for this phase, which gives us a probabilistic guarantee, that each true cluster is represented by at least one of the initial cluster atoms. In the second phase, these cluster atoms are merged in a bottom-up manner resulting in a dendrogram. The final clustering is determined by our objective function. Our experimental evaluation on several real datasets demonstrates that JointClust indeed discovers meaningful and accurate clusterings without requiring the user to specify the number of clusters.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

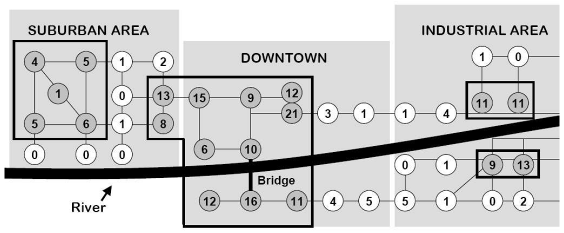


Figure 1: Hotspot example

General Terms

Algorithms

Keywords

Algorithms, Clustering, Graph-Structured Data, Joint Cluster Analysis, Community Identification, Hotspot Analysis

1. INTRODUCTION

Knowledge discovery in databases (KDD) aims at discovering underlying patterns in data. In many applications, attribute and relationship data are available, carrying complementary information. Both types of data are necessary for describing the underlying patterns. For example, in a community identification application, a community is naturally defined by a group of people who share similar interests (defined by attribute data) and are internally connected in terms of social relations (defined by relationship data). In such applications, a joint analysis of both types of data can yield more satisfactory results. [8] proposes a joint cluster model, called Connected k-Center (CkC), to discover k clusters which are cohesive on both attribute and relationship data.

While partitioning clustering algorithms typically require the number of clusters as user input, it is well-known that prior knowledge on the number of clusters is often unavailable in real life applications. Previous research [24, 9, 17, 19] addresses the afore-mentioned problem for classical clustering models considering only attribute data. In this paper, we study the problem of discovering an arbitrary (a-priori unspecified) number of *true clusters* in joint cluster analysis. True clusters are assumed to be compact and distinctive from their neighboring clusters in terms of attribute data and internally connected in terms of relationship data. While in classical clustering methods the neighborhood of clusters is (and can only be) defined in terms of attribute data, in the context of joint cluster analysis it makes more sense to define the neighborhood in terms of relationship data. To motivate this assumption and our approach, we introduce the following two running examples: community identification and hotspot analysis.

Community Identification: A major task in social network analysis is to identify the underlying community structure. In this application, a person can be described by both attribute data, such as demographic information or product preferences, and relationship data, e.g. social relations to other persons. Intuitively, a community is a connected subgraph of people with similar attribute values, which are distinctive of the ones of neighboring communities. In joint cluster analysis, relationship data allows us to naturally define the neighborhood of a cluster. Neighboring communities refer to communities which have members who are directly connected to each other in the social network. Our goal is to identify communities, whose members have similar attribute values, but significantly different values from the ones of the neighboring communities.

Hotspot analysis: In the context of criminology, the goal of hotspot analysis is to identify high-crime areas (*hotspots*) surrounded by areas of relatively low crime rates (*coldspots*). Attribute data represents the frequencies of certain crime types in a particular area over a given period of time. Relationship data represents connectivity information such as a road network. The road network constrains the space in which offenders move and also plays a major role in the appearance of crime attractors and generators like pubs or needle sharing places [23]. Thus, two high-crime areas, which are close to each other using the Euclidean distance, but not directly connected on the road network, have most likely different attractors or generators and should not be merged into one hotspot.

A simplified example with one-dimensional attribute data is illustrated in Figure 1. Nodes are placed in equidistance along the road network, and the attached attribute values measure the number of crimes in the corresponding area. In general, the crime rate in downtown is much higher than in other areas. This means that solely attribute-based clustering methods cannot find the crime hotspot in the suburban area, which has some areas with crime rates that are relatively high compared to their neighborhood, but not especially high from a global point of view. The two parts of the downtown crime hotspot on the different sides of the river are connected via a bridge. When considering not only the attribute but also the relationship data, the hotspot can be correctly identified as one cluster. To conclude, the discussion of the above example applications has motivated the need for finding connected clusters with attribute values significantly different from the attribute values of neighboring clusters.

When discovering an a-priori unspecified number of clusters, the main challenge is to determine a good trade-off between model accuracy and model complexity in an unsupervised fashion. Many existing methods explicitly penalize models with larger numbers of clusters by using an objective function with one term for clustering accuracy and another one for model size (number of clusters), e.g. [24] which uses the Bayesian Information Criterion. Alternatively, objective functions can be used that implicitly penalize large models, e.g. the Silhouette Coefficient [19] that compares for every data object the distance to the assigned cluster center with the distance to the second-closest cluster center. In terms of the algorithmic scheme, we can distinguish *static methods* that compute clusterings for different numbers of clusters from scratch and *dynamic methods* that merge or split one cluster at a time.

In this paper, we explore a dynamic two-phase approach. In the first phase, we identify so called cluster atoms, basic building components that cannot be split in the second phase. The number of initial cluster atoms is chosen in such a way that we have some probabilistic guarantee that every true cluster is represented by at least one of these initial cluster atoms. These cluster atoms are further iteratively refined as input for the second phase. In the second phase, in a dynamic bottom-up approach, cluster atoms are merged step by step. The clustering with the highest objective function value will be returned. Compared to a static method, this dynamic method is more efficient and makes better use of the cluster structures obtained for larger numbers of clusters. As objective function, we employ an adaptation of the Silhouette Coefficient for joint cluster analysis, comparing the distance of a data object to its cluster center against the distance to the centers of all neighboring clusters. This objective function promotes clusters which are compact and as distinctive as possible from their neighboring clusters.

Contributions

The main contributions of this paper are as follows:

1. We introduce and formalize the problem of discovering an a-priori unspecified number of clusters in the context of joint cluster analysis of attribute and relationship data, called *Connected X Clusters (CXC)*.

2. We propose a two-step algorithm (JointClust), producing cluster atoms in the first phase, which are dynamically merged in the second phase.

3. We provide some probabilistic guarantee that every true cluster is represented by at least one of the cluster atoms generated in the first phase.

4. Our experimental evaluation on several real datasets demonstrates that JointClust indeed discovers meaningful and accurate clusterings without requiring the user to specify the number of clusters in advance.

Outline

First, we review the related work in Section 2. Afterwards, we define the Connected X Clusters (CXC) problem in Section 3. A probability analysis, which is used as guidance for the first phase of our algorithm is given in Section 4. We provide a dynamic two-phase algorithm (JointClust) to solve the CXC problem in Section 5. This algorithm is evaluated on several real-world data sets in Section 6. The paper is concluded in Section 7.

2. RELATED WORK

Due to the increasing availability of relationship data, research on graph partitioning has recently attracted much attention [28, 26, 16, 4]. The goal of graph partitioning is to divide a graph into several subgraphs minimizing some objective function, such as cut-size, i.e., the sum of weights of edges crossing between partitions [12, 11]. Similar to our approach are models in which the number of partitions (clusters) is not specified. Methods used for discovering an arbitrary number of graph partitions follow either a top-down or a bottom-up approach. In a top-down approach, in each round one cluster is chosen and partitioned into two sub-clusters until certain conditions are satisfied. Shi and Malik propose a novel global criterion, the normalized cut, as well as a grouping algorithm which recursively segments a graph into subgraphs in [27]. Different criterions, such as Ratio Cut [29, 15], Min-Max Cut [6], Q-function [22, 30], have also been studied for obtaining a natural partitioning of a graph. On the other hand, bottom-up approaches, such as ROCK [14] and CHAMELEON [18], start from assigning each node into its own cluster and merge similar clusters based on different criteria.

Discovering the true cluster structure of attribute data has been studied extensively. Two main approaches, hierarchical clustering methods and density-based methods, are widely used for this purpose. Hierarchical clustering methods, such as single-link, complete-link and average-link [17], CURE [13], BIRCH [31], and XMeans [24], achieve this goal by building a hierarchy of clusters to allow the exploration of clusters at any level of granularity. In particular, XMeans starts from a solution of 2-means and recursively partitions the generated clusters into two sub-clusters until the size of the clustering exceeds a user specified value or none of the possible splits improves a given quality measurement (Bayesian Information Criterion or Akaike Information Criterion). Density-based methods, such as DBSCAN [9], are able to find an arbitrary number of clusters which satisfy the predefined density threshold. The hierarchical density-based method OPTICS [1] does not produce actual clusters and correspondingly cannot identify the number of clusters.

None of the aforementioned methods considers relationship data and attribute data naturally and simultaneously. In graph partitioning, the relationship data can be either original network data or similarity graphs transformed from a purely attribute-based similarity matrix. Some of the graph partitioning approaches operate on weighted graphs, where the weight of an edge is the distance between the attribute values of the linked nodes. However, by transforming nodes' attribute values into edge weights, information contained in attribute data is lost. Ester *et al.* studied a general joint cluster model which considers both attribute and relationship data naturally and simultaneously in [8]. In the paper, the Connected k-Center (CkC) problem is proposed, which is to group data objects into k clusters with the goal of minimizing the maximum distance of any data object to its corresponding cluster center, meanwhile satisfying an internal connectedness constraint on the relationship data. Due to the NP-hardness of CkC, a heuristic algorithm is introduced. Different from the CkC model, the joint cluster model, *Connected X Clusters*, discussed in this paper focuses on the problem of automatically identifying the appropriate number of clusters.

3. PROBLEM DEFINITION

In this section we propose the Connected X Clusters problem, which is a joint cluster model of attribute and relationship data. It does not require the user to specify the number of clusters. First, we define some notations needed in the next sections. Afterwards, we introduce a new criterion for model selection, which considers attribute and relationship data simultaneously. The problem of *Connected X Clusters* is formally defined at the end of the section.

3.1 Definitions

Attribute data can be represented as an $n \times m$ entity-attribute matrix. By applying some similarity measure, an $n \times n$ entity-entity similarity matrix can be calculated to represent pairwise similarities. Relationship data is often modelled by networks comprising nodes and edges. In the context of joint analysis of attribute and relationship data, we adopt the so-called *informative graph* [8] as an unified representation of both data types in this paper. Attribute and undirected relationship data can be transformed into an undirected graph, in which each node represents a data object and each edge a relationship between the data objects associated with the corresponding nodes. The attribute values of a data object are attached to the corresponding node as a weight vector. We formally define an informative graph as follows:

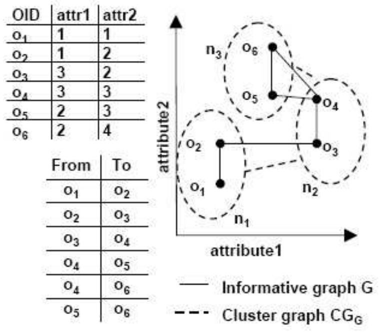
Definition 1 (Informative Graph). *Given a set of n data objects $O = \{o_1, \ldots, o_n\}$. Let $A(o_i)$ be the attribute values of o_i and $R(o_i, o_j) = 1$ iff there is a relationship between o_i and o_j. An **informative graph** is a graph $G = (V, E, w_V)$ with following properties:*

1. *Each vertex $v_i \in V$ corresponds to a uniquely defined data object $o_i \in O$ such that $w(v_i) = A(o_i)$.*
2. *There is an edge between v_i and v_j iff $R(o_i, o_j) = 1$.*

Given attribute and relationship data, the corresponding informative graph will be the input to the algorithm solving the Connected X Clusters problem. A cluster graph (defined below) partitions the node set of the informative graph into internally connected groups (clusters). It is the output of our algorithm. To define the neighborhood of a cluster, we introduce a new term, *G-connected*, such that two clusters are connected, iff they are G-connected.

Definition 2 (G-connected). *Let G_1 and G_2 be two non-overlapping, connected subgraphs of informative graph $G = (V, E, w_V)$. We call G_1 and G_2 $G-$connected, denoted as $G_1 \leftrightarrow_G G_2$ iff there exists at least one edge $\{v_1, v_2\}$ $\in E$, such that $v_1 \in G_1$ and $v_2 \in G_2$.*

Intuitively, the cluster graph of an informative graph is a graph representing the result of a joint cluster algorithm applied to the informative graph. Its nodes represent clusters and contain a set of connected nodes from the informative graph. If the attribute data is two or three dimensional, the visualization of the information graph is straight forward by using the attribute values as coordinates. Before we define formally a cluster graph, we give a simple example in Figure 2, which shows, how to transform attribute and relationship data into an informative graph G, and G into a cluster graph CG_G. The first table contains attribute data, the second relationship data. Note that only symmetric binary relations are considered in this paper. Let us assume the clustering algorithm assigns data object o_1 and o_2 to one cluster, o_3 and o_4 to a second one and the remaining data objects to a third cluster. The resulting cluster graph contains three clusters (nodes): cluster n_1 consists of the data objects o_1 and o_2 ($n_1 = \{o_1, o_2\}$), $n_2 = \{o_3, o_4\}$, and $n_3 = \{o_5, o_6\}$ and two undirected edges from n_1 to n_2 and from n_2 to n_3. Since there is no edge from any node in n_1 to any node in n_3, these nodes are not connected.



OID	attr1	attr2
o_1	1	1
o_2	1	2
o_3	3	2
o_4	3	3
o_5	2	3
o_6	2	4

From	To
o_1	o_2
o_2	o_3
o_3	o_4
o_4	o_5
o_4	o_6
o_5	o_6

Figure 2: Informative and cluster graph example

Definition 3 (Cluster Graph). *Let $G = (V, E, w_V)$ be an informative graph. Let $\{V_1, ..., V_k\}$, $k \geq 2$ be a partition of V, i.e., $V = V_1 \cup ... \cup V_k$ and, $V_i \cap V_j = \emptyset \; \forall 1 \leq i < j \leq k$. A graph CG_G is called* ***cluster graph****, if $CG_G = (\overline{V}, \overline{E})$, where $\overline{V} = \{V_1, ...V_k\}$ and $\overline{E} = \{\{V_i, V_j\} | V_i \leftrightarrow_G V_j\}$ and V_i, $i = 1, ...k$, is internally connected in G (****Connectivity Constraint****).*

3.2 Criteria for Model Selection

There are two general strategies for finding the best cluster model: the static and the dynamic one. A *static* approach computes the clusterings for different numbers of clusters from scratch. It is therefore easy to implement, since one and the same method is called repeatedly with a different number of clusters as input parameter. *Dynamic* methods split or merge one cluster at a time to or from the best current clustering. In most cases, dynamic methods have a runtime advantage, since they do not have to reconsider the whole data set after the splitting or merging but only the relevant part.

Dealing solely with attribute data, the *Silhouette Coefficient* [19] is a measurement, which judges the clustering quality independent of the number of clusters. It compares for every data object the distance to the assigned cluster center with the distance to the second-closest cluster center. Formally, it is defined as $s = \frac{1}{n} \sum_{i=1}^{n} \frac{b(i)-a(i)}{max\{b(i),a(i)\}}$, where $a(i)$ is the distance from data object i to the center of the cluster, to which it was assigned, and $b(i)$ is the distance to the second-closest cluster center.

This definition is not applicable to a relationship and attribute environment with connectivity constraint, since the goal of a joint cluster analysis is not to discover clusters which are as distinctive as possible from *all* other clusters, but only from all *connected* clusters; see also the motivation of our problem definition in Section 3.3. For this reason, we propose an adapted version of the Silhouette Coefficient, called *Joint Silhouette Coefficient*, which incorporates the relationship data. It compares the distance of each data object to the assigned cluster center with the average distance to the centers of all neighboring clusters.

Definition 4 (Joint Silhouette Coefficient). *Given an informative graph $G = (V, E, w_V)$ and a cluster graph $CG_G = (\overline{V}, \overline{E})$, the* ***Joint Silhouette Coefficient S*** *is defined as follows:*

$$S = \frac{1}{|V|} \sum_{i \in V} \frac{b_E(i) - a(i)}{max\{b_E(i), a(i)\}}$$

Let a be the cluster to which i was assigned. $a(i)$ is the distance to the center of a and $b_E(i)$ is the average distance to all connected clusters of a.

For example, in Figure 1 the attribute data of the two hotspots in the industrial area are quite similar. Since they are not connected, they are assigned to different hotspots, which is desirable, since they are very likely to have difference attractors. We do not want to penalize the model for having two clusters (hotspots) with similar values, if the clusters are not connected. Since the Joint Silhouette Coefficient considers only the distance to the neighboring clusters, the existence of these two clusters does not negatively influence the value of the Joint Silhouette Coefficient. Therefore, the Joint Silhouette Coefficient allows us to consider only the *local* neighborhood of a cluster.

3.3 Connected X Clusters Problem

Clustering is the process of partitioning data objects into groups according to some similarity measure, such that data objects within one group are similar to each other and dissimilar to data objects from other groups [20]. This definition is not directly applicable in joint cluster analysis, in which relationships carry important information. In many applications, like social network or hotspot analysis, relationships contain additional information. In these contexts, we require a cluster to be internally connected. The data objects of a cluster are still required to be as similar as possible to each other, but not as dissimilar as possible to *all* clusters, but only to the *connected* clusters. This leads to the following problem definition:

Definition 5 (Connected X Clusters Problem). *Given an informative graph $G = (V, E, w_V)$ and an integer value m, the minimum cluster size. The goal of the* ***Connected X Clusters Problem (CXC)*** *is to find a cluster graph $CG_G = (\overline{V}, \overline{E})$ such that*

1. *CG_G fulfills the* ***minimum size constraint****, i.e. each cluster (node) $v \in \overline{V}$ contains at least m data objects and*

2. *CG_G maximizes the Joint Silhouette Coefficient.*

Intuitively, we want to find a partitioning of the given data, such that each partition is internally connected and larger than the minimum size. We are interested in the partitioning, in which clusters are compact and their cluster centers are as distinctive as possible from the ones of all connected clusters. Our assumption is that the informative graph is connected. If it is not connected, we define the problem separately for each maximum connected component.

In many applications, like social network analysis, a community is expected to have a minimum size. E.g. a scientific community of 10 people might be not very meaningful. The minimum size constraint ensures that the size of each cluster exceeds the minimum size. Note that if desired the minimum size can be set to 1.

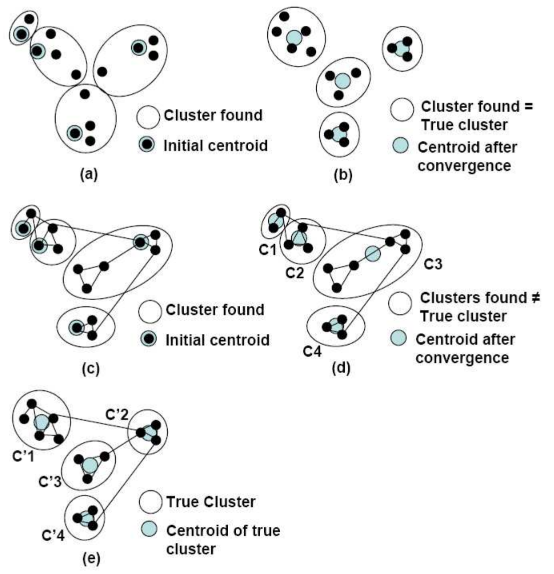


Figure 3: (a) and (b) Influence of initialization on 4-means; (c) and (d) Influence of initialization on JointClust (e) Desired solution of JointClust

4. DETERMINATION OF THE REQUIRED NUMBER OF INITIAL CENTROIDS

The main objective of this paper is to learn the number of clusters in joint cluster analysis. Similar to k-means, our proposed algorithm JointClust also starts with a set of initial centroids. In this section, we will first discuss the influence of the initialization on k-means and on joint cluster analysis. We will see, that the result of joint cluster analysis is crucially dependent on the initial centroids. Afterwards, we determine the appropriate number of initial centroids, so that we can guarantee to have placed at least one initial centroid in each *true cluster* with a (given) high probability. The crucial idea is to pick more than the required number of initial centroids in order to have a probabilistic guarantee that we have at least one initial centroid per true cluster. These centroids together with their neighborhood (as defined in the algorithm) will serve as *cluster atoms*, which are basic building components, that cannot be split in the second phase.

4.1 Influence of the Initialization on k-means

k-means [7, 2] is known to converge to a local minimum of its distortion measurement (total squared distance). That means, the average squared distance can never increase in an iteration step. It is also well-known, that k-means is sensitive to its initialization. Some improvement strategies for finding more sophisticated initializations than random ones were introduced, e.g. by Bradley *et al.* [3]. They suggest to repeatedly subsample and smooth to get a better initialization. However, in most cases k-means is started several times with different initializations and the best one (based on the value of the distortion measurement) is chosen. Figure 3(a) shows an example for a bad initialization, in which 4-means is still able to converge to the desired result; see Figure 3(b).

4.2 Influence of the Initialization on Joint Cluster Analysis

Similar to k-means, our algorithm, JointClust (which will be introduced in the next section) is also very dependent on its initialization. In its first phase JointClust greedily assigns data objects to the initial centroids under consideration of the connectivity constraint. Any joint cluster analysis algorithm considering a connectivity constraint has a very restricted search space due to this constraint resulting in cluster atoms. Therefore, it is more likely to merge distinct clusters together if each true cluster was not represented by at least one initial centroid. As an example, see Figures 3(c) to 3(e). Figure 3(c) shows the informative graph with randomly chosen initial centroids, 3(d) the result after convergence. In Figure 3(e) the desired clustering result can be found. Because there is no direct relationship between between C'2 and C'3 in the true clustering, and the centroids of C2 and C3 in Figure 3(d) are far apart, the centroid of C2 cannot move to split C3 in the desired way.

To overcome this limitation of all joint cluster analysis algorithms, we draw a higher number of initial centroids than the maximal possible number of clusters to guarantee that at least one initial centroid was placed in each true cluster.

4.3 Determination of the Number of Initial Centroids

In the following we propose and prove two theorems. The goal of the first theorem is to show, how small the likelihood is given k clusters and k trials, that we draw exactly one data object from each cluster. The second theorem states how many trials are necessary in order to draw at least one data object per cluster with a certain confidence.

THEOREM 4.1. *Given a data set X. Let $X_1, ...X_k$ be the clustering (partition) of X and $|X_i|$ the number of data objects contained in cluster X_i. The probability that we draw exactly one element of each cluster in k trials is between $(\frac{m}{|X|})^k * k!$ (where m is the minimum size of a cluster) and $\frac{k!}{k^k}$.*

PROOF. Let E_i^k denote the event that we draw at least one data object of cluster X_i in k trials. We are interested in the probability of drawing one data object per cluster in k trials, where the total number of clusters is also k.

$$p = P(E_1^k \wedge ... \wedge E_k^k) = P(\wedge_{i=1}^k (E_i^1)) * k! = \prod_{i=1}^{k} P(E_i^1) * k!$$

If we want to draw at least one data object per cluster (having in total k clusters) and we have only k trials, then this is equivalent to drawing exactly one data object per cluster. The probability for choosing a data object out of an arbitrary cluster is the cluster size divided by the total number of data objects. Therefore $P(E_i^1) = \frac{|X_i|}{|X|}$. In the following we calculate the upper and lower bound, given an arbitrary minimum cluster size $m \leq \frac{|X|}{k}$ using the formula for p.

$$p = k! \prod_{i=1}^{k} \frac{|X_i|}{|X|} \leq \frac{k!}{|X|^k} \left(\frac{\sum_{i=1}^k |X_i|}{k} \right)^k = \frac{k!}{k^k} \quad (1)$$

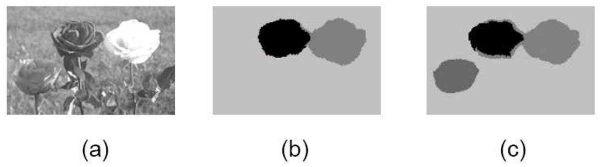


Figure 4: **(a) Original Image (b) Result if no initial centroid was placed in one of the flowers (c) All clusters were represented by at least one initial centroid**

$$p = k! \prod_{i=1}^{k} \frac{|X_i|}{|X|} \geq k! \prod_{i=1}^{k} \frac{m}{|X|} = (\frac{m}{|X|})^k * k! \quad (2)$$

$$(\frac{m}{|X|})^k * k! \leq p \leq \frac{k!}{k^k} \quad (3)$$

In equation (1) we used the the inequality of the arithmetic and geometric mean. Inequality (3) follows immediately from (1) and (2). Therefore, the probability for drawing one data object per cluster, given k clusters, in k trials is between $(\frac{m}{|X|})^k * k!$ and $\frac{k!}{k^k}$, where m is the minimum size of a cluster. □

To illustrate these probabilities, we give the following example. Let $n = 1000$, the minimum size $m = 100$ and $k = 10$, then $p = 0.00036$. Thus, the chances of placing one of the initial centroids in each cluster is very small. In order to address this problem, we propose the following strategy: to find at most k clusters, we use more than k initial centroids. The exact number is determined based on the following theorem.

Required number of initial centroids In Section 5 we will use the following theorem for determining the required number of initial centroids. Therefore, the number of data objects we need to draw, corresponds to the number of initial centroids our algorithm requires.

THEOREM 4.2. *Given a data set X with n data objects and m as the minimum size of a cluster. Let $X_1, ...X_l$ be the clustering (partition) of X and $|X_i|$ be the number of data objects in cluster X_i. Let $k = \lceil \frac{n}{m} \rceil$. In order to choose each cluster at least once with a probability of p, we need to draw at least $s = \lceil k \ln \frac{k}{-\ln p} \rceil$ data objects.*

PROOF. As in the previous proof, let E_i^s denote the event that cluster X_i was chosen at least once within the last s trials. Therefore, $\neg E_i^s$ models that X_i has not been chosen in the last s trials.

$$P(\neg E_i^s) = \left(1 - \frac{|X_i|}{n}\right)^s \leq \left(1 - \frac{1}{k}\right)^s \leq e^{-\frac{s}{k}}$$
$$\iff P(E_i^s) \geq 1 - e^{-\frac{s}{k}}$$

Let $P(E_{all}^s)$ be the probability that all k clusters were chosen at least once after s trials. We have

$$P(E_{all}^s) = P(\wedge_{i=1}^k E_i^s) = \prod_{i=1}^{k} P(E_i^s) \geq (1-e^{-\frac{s}{k}})^k \geq 4^{-ke^{\frac{-s}{k}}},$$

since $\forall x \leq \frac{1}{2} : (1-x)^{\frac{1}{x}} \geq \frac{1}{4}$. In order to hit each cluster at least once in s trials with probability p, we need s trials with $4^{-ke^{\frac{-s}{k}}} \geq p$. By solving the inequality, we get $s = \lceil k \ln \frac{k}{-\ln p} \rceil$, since s has to be an integer. □

Confidence	0.900	0.950	0.990	0.999
Nr of Initial Centroids	61	68	85	108

Table 1: **Number of initial centroids required for $n = 1000$ and $m = 100$**

Note that this theorem is similar to the famous coupon collector problem, e.g. [10]. The values in Table 1 refer to the same example as described before ($n = 1000$ and $m = 100$). Our algorithm JointClust, which will be introduced in the next section, uses Theorem 4.2 in order to determine the required number of initial centroids, so that we have a probabilistic guarantee that JointClust identifies in the first phase the cluster atoms, which are crucial as input for the second phase.

5. ALGORITHM

In this section we introduce JointClust, a dynamic two-phase heuristic algorithm solving the Connected X Clusters Problem. In Section 5.1 we explain JointClust and in Section 5.2 we analyze its runtime.

5.1 JointClust

We propose an algorithm (JointClust) solving the *Connected X Clusters Problem*, which is defined in Definition 5. In summary, JointClust is a bottom-up heuristic algorithm, which consists of two phases. In the first phase, we determine cluster atoms, which are bottom-up merged in the second phase in order to determine the final clustering. Each final cluster consists of at least one cluster atom. In the first phase after the initialization part, we have a probabilistic guarantee that each true cluster is represented by at least one cluster atom. The refinement part improves the quality of the identified cluster atoms. The second phase merges bottom-up the so far identified cluster atoms to find the true clustering based on the Joint Silhouette Coefficient (see Definition 4). We return the clustering with the highest Joint Silhouette Coefficient from the generated dendrogram.

The connectivity constraint restricts the search space for possible clusterings. A top-down strategy would require complete search in each level, which cannot rely on the previous results due to the connectivity constraint. Therefore, we chose a bottom-up approach. The other crucial advantage of this strategy is the runtime, see Section 5.2. We will discuss JointClust now in detail; its pseudo code can be found in Algorithm 1.

The input of JointClust is an informative graph `infGraph`, which can be generated from attribute and relationship data as described in Definition 1, and the minimum cluster size (`minclusterSize`), which is necessary to find meaningful clusters. However, this is not a restriction, since this value can be set to 1. The last parameter is the number of iterations (`nrIt`), which determines how often the refinement part is repeated. The number of iterations is a common parameter in most optimization algorithms, it can be set to a constant value, e.g. 10.

Algorithm 1 JointClust Algorithm

```
1: INPUT: $infGraph$, $minClusterSize$, $nrIt$
2: \\FIRST PHASE - Initialization Part
3: $s$ = reqNrCentr(size($infGraph$)/$minClusterSize$)
4: $iniCentroids$ = $s$ random nodes from $infGraph$
5: $iniClusterAtoms$ = genClusterAtoms($iniCentroids$)
6: $iniClusterAtoms$ =
7: mergeSmallClusters($iniClusterAtoms$)
8: \\FIRST PHASE - Refinement Part
9: $clusterAtoms_0 = iniClusterAtoms$
10: for i = 1 to nrIt do
11:     $centroids_i$ = getMedoids($clusterAtoms_{i-1}$)
12:     $clusterAtoms_i$ = genClusterAtoms($centroids_i$)
13:     $clusterAtoms_i$ = mergeSmallClusters($clusterAtoms_i$)
14: end for
15: \\SECOND PHASE
16: $finalClusterGraph$ =
17: mergeAtoms(clusterGraph($clusterAtoms_{nrIt}$))
```

First Phase

The goal of the first phase is to identify cluster atoms. A *cluster atom* is a basic building component whose contained nodes are connected. It cannot be split in the second phase. The first phase consists of an initialization part and a refinement part. To ensure each true cluster is represented among the candidates we chose a higher number of initial centroids than the maximum number of possible clusters (which is bound by the total number of data objects divided by the minimum size of a cluster). We randomly pick $s = reqNrCentroids(\lceil size(infGraph)/minClusterSize\rceil$ initial centroids (`iniCentroids`) (see Theorem 4.2 for a detailed explanation, as confidence we used a standard value of 0.95) from the input graph. We apply `genClusterAtoms()` to this set.

Cluster Atom Generation (`genClusterAtoms()`) At the beginning, each of the initial centroids (`iniCentroids`) is assigned to its own cluster. We perform a breadth-first search starting from the nodes that are already assigned to clusters. A node is allowed to be assigned to a cluster, if it is directly connected to at least one node contained in this cluster. We greedily assign nodes in the described fashion by always choosing the node that has the smallest distance to the cluster representative.

After having assigned each node to a cluster, the resulting clusters are guaranteed to be internally connected. We construct the cluster graph as follows: for each cluster, we create a new node. Two nodes are connected, if any two of their contained nodes are connected, see also Definition 3. The generated nodes represent cluster atoms.

Since we assume the input graph to be connected, we know that the resulting cluster graph is connected as well. To ensure that the minimum size constraint is fulfilled, we join too small cluster atoms with their closest connected cluster atoms. Function `mergeSmallClusters()` takes care of this.

The initialization part of the first phase guarantees that we have placed at least one centroid (and its corresponding cluster atom) within each true cluster with a certain probability, see Section 4. To improve the quality of the final cluster atoms used as input for the second phase, we iteratively refine the cluster atoms.

The refinement part of the first phase starts with calculating the medoids of the previously discovered cluster atoms. We use these medoids as initial centroids in the `genClusterAtoms()` function. These refined centroids, denoted by `centroids`$_i$, are of better quality than the random ones used in the initialization part of the first phase, since they are centrally located (in terms of attribute and relationship data) in their cluster atoms as we will see in the following. While the initial centroids came with a probabilistic guarantee that at least one of them was chosen from each true cluster, these centroids could have been located on the border (in terms of attribute and/ or relationship data) of a cluster. The refined centroids avoid any of these problems the initial centroids may have. To further improve the quality of the centroids and their corresponding cluster atoms, we recalculate the medoids and restart the `genClusterAtoms()` `nrIt` times. In the following, we explain the improved medoid calculation.

Medoid Calculation (`getMedoids()`) The function `getMedoids()` returns for each cluster atom the node, which minimizes the square distance from all other cluster members to the chosen node. If there are several almost equally good medoid candidates, we select a candidate where most of the nodes in its neighborhood are assigned to the same cluster and have similar values. This strategy guarantees that we do not pick a node surrounded by outliers or a node at the boundary of a cluster.

Second Phase

The result of the first phase is a cluster graph (a set of cluster atoms, which partition the node set of the input graph). We know that each cluster atom is as compact as possible and each true cluster is represented by at least one cluster atom. The goal of the second phase is to find the final clustering by bottom-up merging one pair of similar neighboring cluster atoms in every step, using the Joint Silhouette Coefficient as objective function to choose the best clustering in a particular step.

Merging Strategy (`mergeAtoms()`) The subroutine `mergeAtoms()` (Algorithm 2) has as input a cluster graph. Each edge in the cluster graph represents a possible merge, since it preserves the connectivity and minimum cluster size constraint. At the beginning, the Joint Silhouette Coefficient of the input graph is calculated (Line 3). We calculate the Joint Silhouette Coefficient for all possible merges (edge `e`) of a given cluster graph `clusterGraph`$_k$ and pick the one with the highest value (Line 7). Afterwards, the chosen cluster graph is updated reflecting the merge and thereby the number of clusters is reduced by one (Lines 8 and 9). We continue with this strategy until only two nodes are left. Like the conventional Silhouette Coefficient, the Joint Silhouette Coefficient is a non-monotone function. Therefore, the merging continues until only two clusters are left. The cluster graph with the highest Joint Silhouette Coefficient is returned.

5.2 Runtime Complexity

In this section, we analyze the runtime of JointClust and compare it with the runtime of the completely bottom-up algorithm that starts merging from singleton clusters, containing only one node, instead of from cluster atoms.

First Phase The runtime of `genClusterAtoms()` is $O(s * nrIt * n^2)$, where n is the number of data objects and s the number of initial centroids. Merging too small clusters can be computed in $O(s^2)$, since there are at most s clusters and

Algorithm 2 Sub-routine mergeAtoms()

```
1: INPUT: $clusterGraph_k$
2: //$k = clusterGraph_k$.clusterCount
3: $S_k$ = getJointSilCoeff($clusterGraph_k$)
4: while $k \geq 2$ do
5:   for each edge $e$ in $clusterGraph_k$ do
6:     $S_{k-1,e}$ = getJointSilCoeff($clusterGraph_k$.merge($e$))
7:   end for
8:   $e_{max}$ = edge leading to highest $S_{k-1,e} = S_{k-1}$
9:   $clusterGraph_{k-1} = clusterGraph_k$.merge($e_{max}$)
10:  $k = k - 1$
11: end while
12: return $clusterGraph_i$ with the highest $S_i$
```

each cluster has at most $s-1$ connected neighbors. `nrIt` is a constant. Since $n \gg s$, we consider s as a constant as well, therefore the runtime of the first phase is $O(n^2)$.

Second Phase The number of nodes of the input graph for the second phase is bound by $l = \lceil n/minClusterSize \rceil$. An undirected graph with l nodes has at most $O(l^2)$ edges. In each iteration of the while-loop we try all possible merges of two clusters, i.e. $O(l^2)$. Note that in each iteration the number of clusters decreases by 1, therefore we have at most $O(l)$ iterations of the while-loop. The total runtime of the second phase is therefore $O(l^3)$.

Complete Runtime The run of JointClust is therefore $O(n^2)$, since $l < s \ll n$ and s can be considered as a constant.

Comparison A complete bottom-up algorithm would take $O(n^3)$. JointClust reduces this cubic runtime to a quadratic one.

6. EXPERIMENTAL EVALUATION

We evaluate JointClust on two different types of data sets: social network data and hotspot data (image data). As motivated in the introduction, social network analysis and hotspot analysis require the consideration of attribute and relationship data. Our first data set is a co-author network extracted from DBLP and citeseer. We compare the accuracy of JointClust with the one of X-Means [24] (purely attribute-based) normalized cut [27] (purely relationship-based). (Section 6.1) Due to privacy concerns, we did not have access to any real crime data. Instead, in Section 6.2 we evaluated JointClust on image data, which has a graph structure very similar to hotspot data as described below.

6.1 Social Network

The co-authorship network was generated based on the two well-known scientific literature digital databases citeseer[1] and DBLP[2]. In [21], Newman showed that these databases reflect the properties of general social networks very well. We chose papers, written between 2000 and 2004, belonging to three different research areas: Theory, Machine Learning, and Databases & Data Mining. 1,942 authors were extracted out of these papers. We employed the term frequency inverse-document frequency [25], a state-of-the-art method for text representation, to the 603 most important keywords occurring in the abstracts of the corresponding papers and attached them as attribute data to each author. The co-authorship of at least one paper was used to generate the relationship data for the authors.

[1]http://citeseer.ist.psu.edu/

[2]http://www.informatik.uni-trier.de/ ley/db/

	Accuracy	Identified Number of Clusters
X-Means	61.4%	4
Normalized Cut	68.1%	*
JointClust	76.7%	3

Table 2: Results Social Network - true number of clusters is 3; our implementation of normalized cut required the number of clusters as input, we set it to the true number of clusters (3).

The task was to cluster the authors, based on the keywords of their papers and their co-authorships. In order to evaluate our result, we calculated the accuracy of our unsupervised learning algorithm with respect to the expected result by comparing the original labels of the authors to the assigned ones. Authors were labelled with the majority area of their papers.

We ran the Normalized Cut [27] (provided by [3]), a graph partitioning algorithm and X-Means [24], an attribute-based clustering algorithm, which also does not require the number of clusters as input, on this data set. We used the Java implementation of X-Means provided by Weka 3.5.5[4]. Both of the clustering algorithms used the Cosine Similarity, the standard distance (similarity) function for text documents. JointClust requires the minimum size of a cluster as input parameter, which was set to 100, since scientific communities with less members seem not significant enough. We use the default number of iterations, i.e. 10. X-Means was run with the default parameters, using 2 as minimum number of clusters. Note that X-Means is not able to consider the co-authorship relations, since it can only deal with attribute data.

The results can be found in Table 2. JointClust achieved an average accuracy of 76.7%, whereas X-Means only got 61.4% (both methods averaged over 20 repetitions) and the normalized cut achieved 68.1%. JointClust identified the correct number of clusters in all repetitions, whereas X-Means discovered always 4 clusters (instead of 3) and was in each repetition substantially below the accuracy of JointClust. Our implementation of the normalized cut required the number of clusters as input parameter; we set it to the true number, 3. These results confirm that attribute and relationship data indeed contain complementary information which helps to identify the true clusters.

6.2 Image Data

The goal of *image segmentation* to partition a digital image into into multiple regions (sets of pixels). It has to be distinguished from *object recognition* algorithms which determine if a given set of objects appears in an image. One application of the aforementioned methods are *image retrieval systems*, which aim at finding images that are similar to the query image. In this paper, we use JointClust as a basic image segmentation algorithm, more to demonstrate its application to image data, than to compare ourselves with state-of-the-art approaches in the image processing community. Our main reason to use image data was its similarity

[3]http://www.seas.upenn.edu/˜timothee/software /ncut_multiscale/ncut_multiscale.html

[4]http://sourceforge.net/projects/weka/

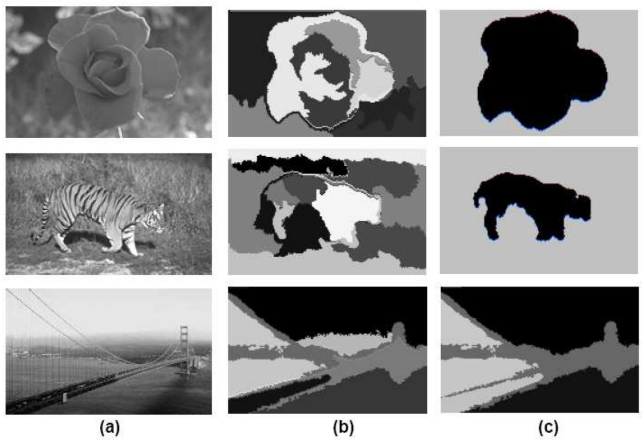


Figure 5: (a) Original Image (b) Result after First Phase (c) Final Result

to hotspot data, to which we did not have access for privacy concerns.

Similarity of Image Data to Hotspot Data

Privacy concerns make the access to crime databases almost impossible. To compensate for the unavailability of this type of data, we run JointClust on image data. On a conceptual level, these two data types are very similar. Most modern North American cities have a grid-like road-network. The resulting graph structure of an image, in which neighboring pixels are connected, is therefore very similar to one of these cities. The crime rate corresponds to the extracted feature values of each pixel. As in hotspot analysis, in image segmentation it makes sense to assume a segment (cluster) to be compact in terms of its attribute values and distinctive from its neighboring clusters.

In this paper, the attribute data was given by the $L^*a^*b^*$ value and three texture features. To extract these values from the input images, we used a MatLab program[5] provided by the university of Berkeley. The details of their feature extraction algorithm and their image retrieval algorithm, blobworld, are described in [5]. Different from their approach, we did not consider the coordinates of the pixels as additional attribute data, but used neighborhood relationships to generate relationship data. These relationships were generated by connecting all pairs of neighboring (directly adjacent) pixels. The goal of blobworld was image retrieval, which is completely different from ours, and a comparison against their method was not meaningful. Unfortunately, the images had no ground truth, so that we could not quantitatively measure the clustering accuracy in this application. Instead, we qualitatively measured the clustering accuracy of selected images by visual inspection.

Technical details: We used the Euclidean distance as distance measurement between the feature vectors. The minimum size of a cluster was set to 1,000 pixels, the total number of pixels per image was 19,712. The number of iterations was set to 10. We evaluated the JointClust algorithm on several images of the famous Corel stock photo collection.

[5]http://elib.cs.berkeley.edu/src/blobworld

The figures in Figure 5 show the results of JointClust on three different images. In part (a) of these figures, a grey-scaled version of the original colored image is shown. In part (b), we present the cluster atoms found in the first phase. The identified number of cluster atoms still exceeds the true number of clusters. However, one can see that each true cluster is already represented by at least one cluster atom. Typically, the second phase succeeds in correctly merging the atoms to detect the true clusters. The last part (c) depicts the final result of the second phase which merges the cluster atoms in a bottom-up manner and chooses the final result based on the value of the Joint Silhouette Coefficient. Note that the CXC model considers only connected regions as clusters. (Due to grey-scale postprocessing of the image, we had to color different clusters with the same shade in order to make the different clusters as recognizable as possible.)

In Figure 5 the rose was very well discovered by JointClust. JointClust also determined the number of clusters correctly. As you can see in part (b) of Figure 5, the first phase produced many cluster atoms, which were correctly merged in the second phase. The rose and tiger image in part (c) of Figure 5 show two clusters. The tiger is completely discovered. All cluster atoms were correctly merged in the second phase. Our model section measure, the Joint Silhouette Coefficient, which only compares neighboring clusters, is able to merge the cluster atoms in the desired way.

Our third image was a bridge (Figure 5). This a very challenging image. The shade is responsible for breaking the bridge into two parts. The connectivity constraint is responsible for connecting the cables with the bridge and splitting the background into two parts. However, even if human beings were asked to partition this image, the answer would not be unanimous: some might merge the background with the cables, some might not. Since the land on the left hand site has colors very similar to the bridge, it was falsely merged by JointClust. In order to recognize such challenging objects, more sophisticated image processing algorithms are necessary, which consider background knowledge such as the shape of certain object types. As mentioned above, the goal of these experiments was not to compete with these methods, but to show further applications of JointClust to real world data, and in particular, to hotspot-like data. Overall, the result of JointClust on the bridge image shows that JointClust is able to recognize connected components.

6.3 Summary

Our experimental evaluation on several real world data sets, a social network and several images, shows that JointClust is able to identify the correct number of clusters and the clusters themselves with a high accuracy. The comparison with X-Means, a clustering algorithm, which cannot consider relationship data, shows that relationship data indeed carry important, complementary information and improve the accuracy significantly.

7. CONCLUSION

It has recently been shown in the literature that joint cluster analysis of attribute and relationship data can greatly improve the clustering accuracy in datasets where attribute and relationship data contain complementary information.

In this paper, we have investigated joint cluster analysis without a-priori specification of the number of clusters, since in many common scenario the true cluster number is not available in advance. We have formalized this problem as the so-called Connected X Clusters (CXC) problem, assuming that true clusters are compact and distinctive from their neighboring clusters in terms of attribute data. Differing from classical attribute-based clustering methods, the neighborhood of clusters is not defined in terms of attribute data but in terms of relationship data. To efficiently solve the CXC problem, we have presented JointClust, an algorithm which adopts a dynamic two-phase approach. JointClust determines cluster atoms in the first phase, which are merged in a bottom-up strategy in the second phase. We conducted an experimental evaluation with real datasets, a social network and several image datasets, showing that JointClust indeed discovers meaningful and accurate clusterings without requiring the user to specify the number of clusters in advance.

Joint cluster analysis is a novel paradigm, and we expect it to become more and more important with increasing number of applications with both attribute and relationship data. This paper suggests several directions for future research. First, we have proposed a distance-based measure, the Joint Silhouette Coefficient, to quantify the trade-off between model complexity and accuracy. Probabilistic measures, with their known advantages and disadvantages, deserve further attention. An alternative top-down algorithm, similar in style to X-means, should be explored and compared to the proposed bottom-up algorithm JointClust. Finally, we plan to evaluate the Connected X Clusters model and the JointClust algorithm further in the mentioned and other real life applications.

8. REFERENCES

[1] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. Optics: ordering points to identify the clustering structure. In *SIGMOD*, 1999.

[2] C. M. Bishop. *Neural networks for pattern recognition.* Oxford: Clarendon Press, 1995.

[3] P. S. Bradley and U. M. Fayyad. Refining initial points for k-means clustering. In *ICML*, 1998.

[4] U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *Algorithms - ESA 2003, 11th Annual European Symposium*, Budapest,Hungary, 2003.

[5] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *VIS*. Springer, 1999.

[6] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cut algorithm for graph partitioning and data clustering. *ICDM*, 2001.

[7] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[8] M. Ester, R. Ge, B. J. Gao, Z. Hu, and B. Ben-Moshe. Joint cluster analysis of attribute data and relationship data: the connected k-center problem. In *SDM*, 2006.

[9] 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*, 1996.

[10] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. New York: Wiley, 1968.

[11] P. Fjallstrom. Algorithms for graph partitioning: A survey. *Linkoping Electronic Articles in Computer and Information Science*, 1998.

[12] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete problems. In *Sixth annual ACM symposium on Theory of computing*, 1974.

[13] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. In *SIGMOD*, 1998.

[14] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. In *Information Systems*, 2000.

[15] L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. on Computed Aided Desgin*, 1992.

[16] R. A. Hanneman and M. Riddle. *Introduction to social network methods.* http://faculty.ucr.edu/∼hanneman/, 2005.

[17] A. Jain and R. Dubes. *Algorithms for clustering data.* Prentice Hall, 1988.

[18] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 1999.

[19] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* New York: Wiley, 1990.

[20] T. M. Mitchell. *Machine Learning.* New York; London: McGraw-Hill, 1997.

[21] M. Newman. The structure of scientific collaboration networks. In *Proc. Natl. Acad. Sci. 98*, 2001.

[22] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review*, 2004.

[23] A. Okabe, K.-I. Okunuki, and S. Shino. The sanet toolbox: New methods for network spatial analysis. In *Transactions in GIS*. Blackwell Publishing, July 2006.

[24] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, San Francisco, 2000.

[25] G. Salton and M. J. McGill. *Introduction to modern information retrieval.* McGraw-Hill, 1983.

[26] J. Scott. *Social Network Analysis: A handbook.* Sage Publications, London, 2000.

[27] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[28] S. Wasserman and K. Faust. *Social Network Analysis.* Cambridge University Press, 1994.

[29] Y. Wei and C. Cheng. Towards efficient hierarchical designs by ratio cut partitioning. In *IEEE Conf. on Computer-Aided Design*, 1989.

[30] S. White and P. Smyth. A spectral clustering approach to finding comm. in graphs. In *SDM*, 2005.

[31] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *SIGMOD*, 1996.