Graph Mining and Communities Detection

Etienne Cuvelier and Marie-Aude Aufaure

Ecole Centrale Paris,
MAS Laboratory, Business Intelligence Team
Grande Voie des Vignes
F-92 295 Châtenay-Malabry Cedex, France
`cuvelier.etienne@ecp.fr`, `marie-aude.aufaure@ecp.fr`

Key words: Graph Mining, Community Detection, Data Mining

Abstract. The incredible rising of on-line social networks gives a new and very strong interest to the set of techniques developed since several decades to mining graphs and social networks. In particularly community detection methods can bring very valuable informations about the structure of an existing social network in the Business Intelligence framework. In this chapter we give a large view, firstly of what could be a community in a social network, and then we list he most popular techniques to detect such communities. Some of these techniques were particularly developed in the SNA context, while other are adaptations of classical clustering techniques. We have sorted them in following an increasing complexity order, because with very big graphs the complexity can be decisive for the choice of an algorithm.

1 Introduction

In the actual interconnected world, and the rising of online social networks the graph mining and the community detection become completely up-to-date. If the interest for social networks analysis exists, even in germ, from the beginning of sociology, in the works of Emile Durkheim or Auguste Comte, a more systematic study started in the 30's with the work of Moreno, Davis and Mayo (see [1] and [2] for a more complete state of the art of the social network analysis). And this systematic study includes the use of graph theory. The theory of graphs exists since Euler's solution of the Knigsberg's bridges problem in 1736, but networks, treated in the pioneer times of Euler (for graph theory) and Moreno (for social network analysis), contained only some dozens nodes at most, while the rising of computer science and one of its fields, the data analysis (see [3] or [4] for a good introduction this latter), allow to analyze graphs of big sizes, and permit to develop one of the task of social network analysis: finding groups with high concentrations of relations within the group and low concentration between these groups, which is called community detection [5], [6] [7].

This overview of the community detection algorithms is organized as follow : in section 2 we introduce the basics of social network and network theory. In section 3 we give a review of the definition of a cluster or community in social network analysis. Then in section 4 we take a look to the set of measures which can be used in clustering of social networks. Section 5, 6, 7 and 8 give a review of the most important clustering methods. The order chosen for this review is directly linked to the complexity : we begin with the "cheapest", and then more scalable (partitional and hierarchical methods), then we continue with a more greedy but very efficient technique : the spectral clustering, and we end with the Galois lattices which are costly methods, but with very rich results. Then we close this part with a discussion.

2 Social Networks

Even if Moreno [8] was the first to use points and lines to represent social configurations, it was Cartwright and Harary [9] which made the link with the graph theory, and thus introduced the actual graph representation of social network (see [10] for a review of the evolution of social networks representation): individuals are represented using points, called *nodes* or *vertices*, and social relationships are represented using lines, called *edges* or *links*, between nodes. In figure 1 we show an example of

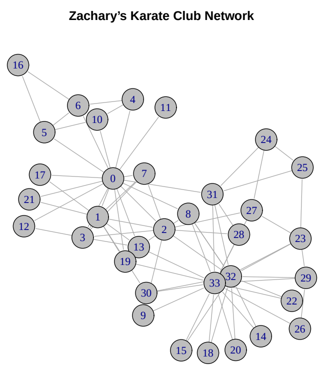

Fig. 1. A classical example of social network : the Zachary's Karate Club.

social networks using this graph representation: The first one (fig. 1) is a the well-known graph [11]: the Zachary's Karate Club, it consists in 34 vertices, the members of a karate club in the United States, who were observed during a period of three years. Edges connect individuals who were observed to interact outside the activities of the club.

Let use introduce here the basic concepts of graph theory. A *graph* G is defined as a pair of sets: $G = (V, E)$, where V is the set of *vertices* or *nodes*, and E the set of *edges* or *links* which connect vertices. The two vertices connected by an edge are called *endpoints* of this latter. Conversely the edges connecting a vertice to the other vertices are called *incident* edges of the node. The number of vertices $|V| = n$ is called the *order* the graph, while the number of edges $|E| = m$ is called the *size* of the graph. If $|E| = n(n-1)/2$, i.e. if any pair of vertices are connected, then the graph is called *complete* .

A graph can be *directed* or *undirected*. In the first case, also called *digraph*, an edge is denoted as pair (v, w), where v is the origin and w the target, and in the social networks framework it means: "v is in relation with w", the opposite being true if and only if there exists an edge (w, v). If the graph G is undirected, then an edge is denoted as the set of its vertices : $\{v, w\}$. For the sake of simplification, when we will give a definition valid in both cases, we will use this last notation.

Most of the times, vertices are labeled, but it can also be the case for edges, then G is called a *labeled graph*. Moreover, if exists a function $\omega : E \rightarrow \mathbb{R}$ that assigns a weight for each edge, then G is a *weighted* graph.

Two vertices v and u are called *neighbors* or *adjacent*, if they are connected by an edge. The set of neighbors of a node v, denoted $\Gamma(v)$, is called its *neighborhood*.

The topology of the graph can be captured in the *adjacency matrix* $A = (a_{i,j})$: where

$$a_{i,j} = \begin{cases} 1 \text{ if } (v_i, v_j) \in E, \\ 0 \text{ otherwise.} \end{cases} \tag{1}$$

If G is a weighted graph, then $a_{i,j} = \omega(v_i, v_j)$ and in this case we prefer to use the notation $W = (w_{i,j}) = (\omega(v_i, v_j))$. Of course in an unweighted graph $\omega(v_i, v_j) \in \{0, 1\}$. For an unweighted graph, the *degree* of a vertice v, denoted $deg(v)$, is defined as the number of incident edges, but a more general definition is given using the weights matrix W:

$$deg(v_i) = \sum_{j=1}^{n} w_{i,j}. \tag{2}$$

A *subgraph* $G' = (V', E')$ of a graph $G = (V, E)$ is such $V' \subset V$, $E' \subset E$ and $\{v, u\} \in E'$ implies $v, u \in V'$. The graph G is a *supergraph* of G'. A subset C of V can define an *induced subgraph* $G(C) = (C, E(C))$, where

$$E(C) = \{(v, u) \in E | v, u \in C\} \tag{3}$$

A complete subgraph is called a *clique*.

The *density* of a subgraph $C(V(C), E(C))$ is the ratio between $|E(C)|$ and the maximum possible number of edges:

$$\delta(G(C)) = \frac{|E(C)|}{|V(C)|(|V(C)| - 1)/2} \tag{4}$$

this definition still being valid for the whole graph G.

A partition of the vertices set V in two subsets C and $V \setminus C$ is called a *cut*. The *cut size*, denoted $c(C, V \setminus C)$ is the number of edges of G joining vertices of C with vertices of $V \setminus C$:

$$c(C, V \setminus C) = |\,\{\{u, v\} \in E | u \in C, v \in V \setminus C\}\,| \tag{5}$$

Both set C and $V \setminus C$ define the cut, but usually the cut is identified by the smaller one.
A *path* between two vertices v and u in a graph G is a sequence of edges starting with $v_0 = v$ an such the last one is $v_k = u$:

$$\{v, v_1\}, \{v_1, v_2\}, \cdots, \{v_{k-1}, u\}. \tag{6}$$

A path P is also a subgraph of G: $P = (V(P), E(P))$ with $V(P) = \{v_0, \cdots, v_{k1}\}$ and $E(P) = \{\{v_0, v_1\}, \{v_1, v_2\}, \cdots, \{v_{k-1}, v_k\}\}$. The *length* of a path is the number of edge in this path, k in expression (6). A *shortest path* or *geodesic* between two vertices is a path of minimal length, and the distance between two vertices is the length of a geodesic between these two vertices. The *diameter* of a graph is the maximal distance that can be found between two nodes.
If no vertice is repeated, then the path is *simple*. If there exists a path between two vertices v and u, they are *connected*. The graph is also called a *connected graph* if for any pair of vertices v and u, there is, at least, one path connecting v and u. Otherwise, i.e. if there is some vertices which cannot be reached from other, then the graph is *disconnected*. If there is two nodes without path between them, then there is, at least two connected subgraphs, and a maximal connected subgraph is called a *connected component*. The *edge connectivity* of a graph G is the minimal number of nodes to be removed so that G is disconnected, and is denoted $k(G)$.
A *cycle* is a path such the first and the last node are equal, i.e. $v_0 = v_k$. A graph without cycle is called a *forest* or an *acyclic graph*, and a connected forest is a *tree*. The edge connectivity $k(T)$ of a tree T is equal to one, because, if the tree contains n vertices, it had $n - 1$ edges, and if any of these is removed, the tree is divided in two disconnected trees.
A connected acyclic subgraph $G' = (V', E')$ such $V' = V$, i.e. all the vertices of G are also in G', is called a *spanning tree*. Every connected graph contains a least a spanning tree. For weighted graphs, a *minimum spanning tree* is the spanning tree such the sum of the weights of the edges is minimal. We can define a *maximum spanning tree* similarly.

3 Community Definitions

In the clustering framework a *community* is a cluster of nodes in a graph [12], but a very important question is *what is a cluster?* Even in the clustering literature there is non complete agreement between all authors (see [13] for a review), but most of the time, the objects in a cluster must be more similar than objects out of this cluster: *the objects are clustered or grouped based on the principle of maximizing the intra-class*

similarity and minimizing the inter-class similarity. Let us remark that, this definition implies the necessity to define the notions of similarity measure and/or cluster fitness measure.
In the graph framework, we can agree that the goal of clustering is to divide the data set into clusters, such nodes of a cluster must be more connected with nodes of this cluster, than with nodes outside of the cluster [7] and [6]. It implies that it must exists at least a path between two nodes of a cluster, and this path must be internal to the cluster.
Then the transposition of the above definition of a cluster into the graph context could be the following: *the vertices are clustered or grouped based on the principle of maximizing the intra-class connections and minimizing the inter-class connections.*
There is several manner to quantify these internal and external connections of a cluster C. A first one is to divide the degree of a vertice in two parts: the internal degree $deg_i(v, C)$ and the external degree $deg_e(v, C)$:

$$deg_i(v, C) = |\Gamma(v) \cap C| \tag{7}$$

$$deg_e(v, C) = |\Gamma(v) \cap (V \setminus C)|. \tag{8}$$

And, as $\Gamma(v) = \{\Gamma(v) \cap C\} \cup \{\Gamma(v) \cap (V \setminus C)\}$

$$deg(v) = deg_i(v, C) + deg_e(v, C). \tag{9}$$

Of course, if $deg_e(v, C) = 0$, then $v \in C$ is surely a good assignation for v, and conversely if $deg_i(v, C) = 0$, then we must have $v \notin C$.
The internal and external degrees, can be seen as the “vertice’s point of view” of indicators of the belonging to a cluster. For a “cluster’s point of view”, we must take a look at the notion of graph density. The *intra-cluster density* $\delta_i(C)$ and the *inter-cluster density* $\delta_e(C)$ are adapted versions of the density of a subgraph defined by expression (4), the first one being the quotient of the number of internal edges of C and the maximal possible number of internal nodes:

$$\delta_i(C) = \frac{|\{\{u, v\} | u, v \in C\}|}{|C|(|C| - 1)/2} \tag{10}$$

and the second one being the result of the number of edges with one vertice inside C, and the other outside of C, divided by the maximal possible number of edges in this configuration:

$$\delta_e(C) = \frac{|\{\{u, v\} | u \in C, v \notin C\}|}{|C|(|G| - |C|)}. \tag{11}$$

Of course for a given cluster C we expect $\delta_i(C)$ and $\delta_e(C)$ to be substantially, respectively, larger and smaller than the average density $\delta(G)$. And, in a “partition’s point of view”, the internal density of the partition, given by the sum of densities over all the clusters, must be appreciably higher than the density of the graph $\delta(G)$.
The comparison of the density of inner ties versus the average density of the graph, leads to define a community in comparison to the rest of the graph, but a community can be considered independently of the graph as a whole. Local definitions of communities [6], focus only on the cohesion

of the studied subgraph, including possibly its direct neighborhood, but ignoring the rest of the graph. In [2] Wasserman identify four criteria to define a cohesive subgroup: *complete mutuality*, *reachability*, *nodal degree*, *internal versus external cohesion*.
The concept of *complete mutuality* states that, in a very strict sense, in a community, all member of a subgroup must be “friends” with all members of the subgroup. In graph theory, it corresponds to a clique [14].
But the definition of a community as a clique is very too strict that is why relaxed definitions of the notion of clique leads to the *reachability*. An *n-clique* (or *k-clique*) [15] is a maximal subgraph such, for any pair of vertices, there exists at least a geodesic no larger than n (or k). The classical clique is then a 1-clique. But a geodesic path of an n-clique could run outside of this latter, and then the diameter of the subgraph can exceed n. That is why the notion of *n-club* and *n-clan* was suggested [16]. An *n-clan* is an *n-clique* with diameter not larger than n, while an *n-club* is a maximal subgraph of diameter n.
The use of the *nodal degree* to define a community imposes a constraint on the number of adjacent vertices. A *k-plex* [17] is a maximal subgraph such each vertice is adjacent to all other vertices of the subgraph except for k of them. Conversely a *k-core* is a maximal subgraph such each vertice is adjacent to, at least, k other vertices of the subgraph [18].
Finally, comparing *internal versus external cohesion*, an *LS-set* [19], or *strong community* [20] is a subgraph C such for each node $v \in C$ we have $deg_i(v, C) > deg_e(v, C)$.
Another point of view to define a community is to say that the number of ties with the outside of the community must be low. That leads to use the notion cut size to define a cluster, and to try to minimize it. Rather than using directly the cut size, the conductance [21] $\Phi(C)$ of a community C is defined to taking into account the order of the cluster and the outside of the cluster:

$$\Phi(C) = \frac{c(C, V \setminus C)}{\min\{deg(C), deg(V \setminus C)\}} \tag{12}$$

where $deg(C)$ and $deg(V \setminus C)$ are the total degrees of C and of the rest of the graph. The minimum of the conductance is obtained when a C community have a low cut size and when the total degree of the cluster and its complement are equal [22].
The cut size is sometime also called the *external degree* of a community C

$$deg_e(C) = |\{\{u, v\} \in E | u \in S, v \in V \setminus S\}| \tag{13}$$

while the *internal degree* is defined by

$$deg_i(C) = |\{\{u, v\} \in E | u, v \in S\}| \tag{14}$$

and then $deg(C) = deg_i(C) + deg_e(C)$. Internal and external degrees are used to define the *relative density* [23]:

$$\rho(C) = \frac{deg_i(C)}{deg(C)}. \tag{15}$$

A community with strong inner ties must have a higher ρ.
Another idea to define a community, is given using the possible flow of information. In a group, given two nodes, at least one shortest path between these nodes, passing through the edges of the group must exist. Conversely if an edge is on many shortest path between several nodes, we can suppose that it is a connection edge between two communities. Finding the connecting edges permits to find the connected communities. It is the idea of the *betweenness*, and more precisely the *edge (or site) betweenness* introduced by [24]. The figure 2 gives an illustration of a node with a maximum edge betweenness.

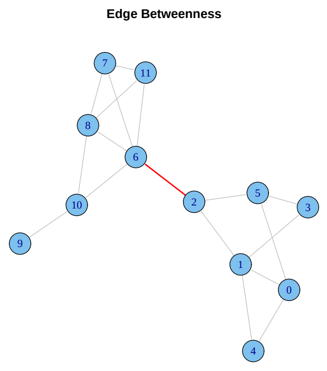

Fig. 2. An example of maximum edge betweenness (edge between nodes 2 and 6).

Finally, the notion of *edge connectivity* k is also used to define a community: an *highly connected subgraph (HCS)* [25] is a subgraph $C \in G$ such

$$k(C) > \frac{n}{2}. \qquad (16)$$

4 Measure for Clusters

Most of the clustering algorithms are based on one or several measures, to be optimized and/or to be used to compare different cluster affectations. We are going to give here the most popular ones.
If we can embed the graph into a n-dimensional Euclidean space, then we can use the classical distances like *Euclidean distance*, *Manhattan distance* or *cosine similarity*, but this embedding into a n-dimensional space can be seen as an artificial construction, and then a distance defined in such a space, used on a graph, is subject to the same criticism. It can be more suitable to work directly with informations included in the adjacency matrix, defining a distance based on this latter [26],[2]:

$$d_{i,j} = \sqrt{\sum_{k \neq i,j} (a_{i,k} - a_{j,k})^2}. \tag{17}$$

This dissimilarity measures the structural equivalence [27]: two vertices are structurally equivalent if they have the same neighbors, which is the case when $d_{i,j} = 0$.
Another measure directly defined on the adjacency matrix is the Pearson correlation matrix computed on rows or columns of A:

$$c_{i,j} = \frac{\sum_{k=1}^{n} (a_{i,k} - \mu_i)(a_{j,k} - \mu_j)}{n \sigma_j \sigma_j} \tag{18}$$

with $\mu_i = \sum_k a_{i,k}/n$ and $\sigma_i = \sqrt{\sum_k (a_{i,k} - \mu_k)^2/n}$.
Another popular seed to build (dis)similarity measure is the *Jaccard index* which measures similarity between sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}. \tag{19}$$

A first use of the Jaccard index in the graph theory context is to measure the *overlap* of the neighborhoods of two nodes v and u:

$$\omega(v,u) = \frac{|\Gamma(v) \cap \Gamma(u)|}{|\Gamma(v) \cup \Gamma(u)|} \tag{20}$$

which is equal to zero when there is no common neighbors, and one when v and u are structurally equivalent.

5 Partitional Clustering

Partitional algorithms try to find a partition of a set of data, with a given number of cluster equal to k. The "best" partition is searched using jointly, most of the times, a distance or dissimilarity measure and a quality criterion of the found partition.
The most popular partitional algorithm (with several variants) is the *k-means clustering* [28]. We give here its most general form. The algorithm try to find a partition $\mathcal{C} = (C_1, \cdots, C_k)$ of a set of n objects denoted $\{x_1, \cdots, x_n\}$, which minimize the *sum-of-square (SSQ)* criterion[1]:

$$g_n(\mathcal{C}) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, c_i) \tag{21}$$

where $d(x, c_i)$ measures the dissimilarity between the object x and the *centroid* of the class C_i. Expression (21) is called SSQ, because initially $d(x, c_i)$ was computed in the following way : $||x - c_i||^2$.
The general algorithm is the following :
Starting step: Determine an initial vector $(c_1^{(0)} \cdots, c_k^{(0)})$,

[1] The SSQ is also termed inertia, squared error function, variance criterion, or trace criterion.

```
Repeat until stationarity: $t \leftarrow t + 1$
    Assignment step: Assign each observation to the cluster with the
        closest centroid:
            $$C_i^{(t+1)} = \left\{ x : d(x, c_i^{(t)}) \leq d(x, c_j^{(t)}) \text{ for all } j = 1, \dots, k \right\} \quad (22)$$
    Update step: Compute the new centroids $(c_1^{(t+1)}, \cdots, c_k^{(t+1)})$ of the
        clusters.
```

Originally centroids of the clusters were computed as the means of the clusters but a more robust version can use the median of the cluster, and [29] have proposed to use as "centroid" (then called prototypes) the most typical subset (pair, triple,...) of objects from this class. Most of the times the starting partition is chosen randomly.

Even if k-means algorithms are efficient, they have several drawbacks

– the choice of k: the number of clusters k is an important input parameter, and an inappropriate choice may leads too non significant results,
– spherical clusters: the algorithm tends to output spherical data, and then works well only when spherical clusters are naturally available in data,
– instability: the random starting partition can leads to a local optimum for the criterion function, and this local optimum can change significantly with another initial partition; to decrease the effect of this instability, a solution it to execute the algorithm several times with different random starting partitions and to retain the solution with the best criterion value.

The complexity of such partitional algorithms is given by $O(m^2 \times n \times k)$ [4] where m is the number of attributes, n the number of objects to cluster and k the number of clusters.

6 Hierarchical Algorithms

Hierarchical clustering algorithms are divided into two types, depending on whether the partition is refined or coarsened during each iteration:

– *agglomerative algorithms* which start with a set of small initial clusters and iteratively merging these clusters into larger ones,
– *divisive algorithms* which split the dataset iteratively or recursively into smaller and smaller clusters.

At each step, the clustering algorithm musts select the clusters to merge or split by optimizing a quality criterion.

6.1 Agglomerative Algorithms

The idea agglomerative algorithms is simple: at the starting point, the n objects to cluster $\{x_1, \cdots, x_n\}$ are their own classes: $\{\{x_1\}, \cdots, \{x_n\}\}$, then at each stage we merged the two more similar clusters.

A dissimilarity measure D between two clusters A and B is mandatory, and for a given dissimilarity measure d between objects, several D exist, we give here the most popular:

- the single linkage: $D(A, B) = \min\{d(x, y) : x \in A,\ y \in B\}$,
- the complete linkage: $D(A, B) = \max\{d(x, y) : x \in A,\ y \in B\}$,
- the average linkage: $D(A, B) = \frac{1}{|A| \cdot |B|} \sum_{x \in A} \sum_{y \in B} d(x, y)$.

In [30] the Lance formula was given to compute $D(C, A \cup B)$ given $D(C, A)$,$D(C, B)$ and $D(A, B)$. The result of the clustering process is shown in a *dendrogram*, like in figure 3. In this kind of graph, mergers of communities are drawn like horizontal lines, and the height of this line represents the dissimilarity between the merged clusters (or objects). Thus, in figure 3, individuals 24 and 25 are closest than individuals 23 and 27. For a given k it is easy to find a partition of objects, because an horizontal cut of the dendrogram gives one single partition, we have only to choose the corresponding height. In figure 3 cuts A, B and C, give respectively 4, 7 and 10 clusters. The dendrogram is a hierarchical structure because each community is fully included in a community of higher level.

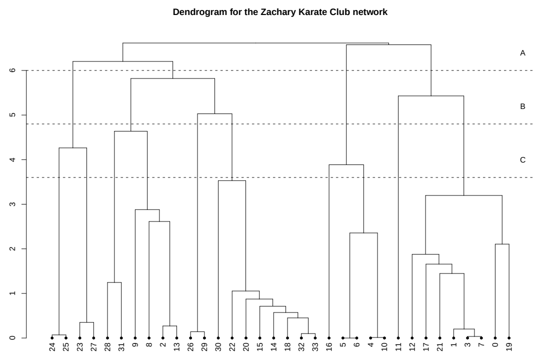

Fig. 3. A dendrogram for the Zachary Karate club network.

The drawbacks of agglomerative hierarchical methods are the following: firstly, vertices of a community may be not correctly classified, and some nodes could be missed even if they have a central role in their cluster [31]; secondly the computational complexity is $O(n^2)$ for the single linkage and $O(n^2 \log n)$ for complete and average linkages, then very big datasets are to avoid. A solution to decrease this computing time is to apply a k-means with a relatively high k, but with $k << n$, and apply the hierarchical clustering on the previous results.

6.2 Divisive Algorithms

Betweenness Using the notion of betweenness [24], it is possible to use a divisive approach to detect communities. The algorithm given in [12] and [5] splits the network into clusters by removing, step after step, edges with the higher betweenness value. In the algorithm, the two following steps are repeated:

1. compute the edge betweenness for all edges of the running graph,
2. remove the edge with the largest value (which gives the new running graph).

Of course, the immediate question is: when to stop this iterative algorithm? Having a criterion to improve at each step should permit to stop when no improvement is gained with an iteration. Such a criterion is proposed in [31]: the *modularity*. Modularity is normally defined for weighted graphs, but can be applied on unweighted graph using $\omega(u,v) = 1$ for all edges. There is many formulation of the modularity, but we give the one found in [7]. The modularity of a partition $(C_1, \cdots, C_k)$ is defined by

$$\mathcal{M}(C_1, \cdots, C_k) = \sum_{i=1}^{k} \varepsilon_{i,i} - \sum_{i \neq j} \varepsilon_{i,j} \tag{23}$$

with

$$\varepsilon_{i,j} = \sum_{v \in C_i, u \in C_j} \omega(v,u) \tag{24}$$

where each edge is included at most once in the computation.
Let us remark that the modularity is directly linked to internal and external degrees (cf. expressions (14) and (13)):

$$deg_i(C_i) = \varepsilon_{i,i} \ \& \ deg_e(C_i) = \sum_{i \neq j} \varepsilon_{i,j}. \tag{25}$$

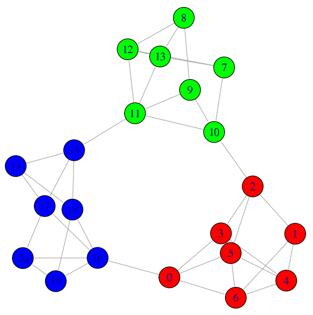

Fig. 4. A simple example of clustering based on betweenness.

The quickest algorithms can compute the edge betwneenness on a graph in $O(n \cdot m)$ for unweighted graphs and in $O(n \cdot m + n^2 logn)$ for the weighted version [32], using for each vertice a single source shortest path problem using a modified version of Dijkstra's solution or a breadth-first search.
Figure 4 gives an illustration of communities found using the edge betweenness.

Cuts Several community detection's algorithms are based on the minimization of the number of edges which link the clusters. These algorithms are, more or less, based on the minimization for each cluster of the cut size between the cluster and the outside of this latter. The *Kernighan-Lin algorithm* is one the first algorithm [33] in this way. These authors worked on the problem of partitioning electronic circuits onto boards, and the need of minimizing the number of connections between the cards. The motivation comes from the fact that there is a limit to the number of components on a card and, connections between cards have an high cost. They use, as quality criterion for the partition, the difference between the internal and the external degree of a cluster:

$$Q(C) = deg_i(C) - deg_e(C) \tag{26}$$

and they try to find the bi-partition of the graph which maximizes Q. To find the best subdivision in two clusters, they start with an arbitrary subdivision, compute Q, and then, subsets consisting of equal numbers of vertices are swapped between the two clusters to find when Q increases. The time execution is in $O(n^2 \log n)$. If we want to find k clusters, the procedure is repeated $k-1$ times on the whole set.
Rather than using directly the cut size, another popular criterion for hierarchical divisive clustering on graph is the conductance (12), because it take into account the order of the sets that are being cut apart [7].
The two main problems with these divisive algorithms, is firstly the fact that find the bisection of the graph with the minimal cut size is an NP-complete problem [34] and find a cut wit the minimal conductance is NP-hard [35]. The second problem is shared by most hierarchical divisive algorithms: when to stop splitting the graph? One can fix the number of clusters at the beginning of the process, but other approaches exist like in [25], where authors propose to use the edge connectivity and the notion of highly connected subgraph (HCS) (see (16)): they do not split a subgraph which is also an HCS.

7 Spectral methods

Spectral algorithms constitute a very particular class of techniques. And this particularity is to perform the classification based on eigenvectors of matrix build upon the adjacency (or weight) matrix.
In this part of the document, we suppose that G is an undirected, weighted graph, with positive symmetric weights matrix W (cf. expression (2)): $w_{i,j} = w_{j,i} \geq 0$. Moreover we need to define the *degree matrix* D:

$$D = W \cdot \mathcal{I} \tag{27}$$

where $\mathcal{I}$ is the identity matrix. D is such that we found the degrees $deg(v_i)$ on the diagonal. Now, we are able to define the spectral clustering: the *Laplacian matrix*:

$$L = D - W. \tag{28}$$

The more interesting property of L is directly linked to the notion of connected component, which is a connected subgraph A without connection with $V \setminus A$. Let us suppose that there is exactly k connected components in G, which form a partition of $G : C_1, \cdots, C_k$, with $c_i = |C_i|$. Without loss of generality, we can assume that the vertices can be ordered according to their connected component. Then, the adjacency matrix W has a block diagonal form, and L too:

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{pmatrix} \tag{29}$$

where each of the blocks L_i is a square matrix, of order c_i, which is the Laplacian of the connected component C_i. As for all block diagonal matrix, we know that the spectrum of L, e.g. the list of its eigenvalues denoted $\sigma(L)$, is equal to the union of the spectrum of the L_j. Moreover it can be shown [36] that the smallest eigenvalue of L is 0 and its multiplicity is exactly equal to k, the number of connected components. Let us recall the classical similarity relation $L = Q\Lambda Q^{-1}$. And as L is a block diagonal matrix, Q and Λ too.

Now, let us suppose, that in each spectrum matrix Λ_i, eigenvalues are sorted in the decreasing order: $\lambda_{i_1} \geq \lambda_{i_2} \geq \cdots \geq \lambda_{i_{c_i}}$, where the last eigenvalue $\lambda_{i_{c_i}}$ is 0. Using expression (27) it is easy to show that the constant one vector $\mathbf{1}_{c_i} = (1, \cdots, 1)^t$ of dimension c_i is an eigenvector of this last eigenvalue:

$$L_i \cdot \mathbf{1}_{c_i} = D_i \cdot \mathbf{1}_{c_i} - W_i \cdot \mathbf{1}_{c_i} = \begin{pmatrix} d_1 \\ \vdots \\ d_{c_i} \end{pmatrix} - \begin{pmatrix} \sum \omega_{1,j} \\ \vdots \\ \sum \omega_{c_i,j} \end{pmatrix} = 0 \cdot \mathbf{1}_{c_i}. \tag{30}$$

As a direct consequence of this, the last vector column of Q_i is the constant one vector $\mathbf{1}_{c_i}$, and then the corresponding column in Q has the following form:

$$(\underbrace{0, \cdots, 0}_{\sum_{j=1}^{i-1} c_j}, \underbrace{1, \cdots, 1}_{c_i}, \underbrace{0, \cdots, 0}_{\sum_{j=i+1}^{k} c_j})^t. \tag{31}$$

These last column vectors of each block of Q are indicator vectors of the k connected components: there is a one only on the rows corresponding to indices of the nodes of the concerned connected component. Then if we denote $\mathbf{u_1} \cdots, \mathbf{u_k}$ these indicator vectors and build $U \in \mathbb{R}^{n \times k}$ with these latter, and denote $Y = U^t$ the transposition of U, for which columns vectors y_i belong to $\mathbb{R}^k$, then y_i contains only zeros except a one for the dimension corresponding to its connected component, i.e. if $v_i \in C_k$, then $y_{j,i} = 1$ if $j = k$, and $y_{j,i} = 0$ otherwise.

But define a community as a connected component is rather strict, and the introduction of one edges between two connected components causes some eigenvalues that were zero become slightly larger than zero but the underlying structure can be seen using the eigenvectors of the Laplacian and easily retrieved using a simple k-means on Y. In the general case, it remains to know k, but it is a problem shared with almost all clustering algorithms. Then, given a Laplacian L and a number of cluster k, the general algorithm for spectral clustering is the following:

1. compute the eigenvalues and sort them such $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$,
2. compute the last k eigenvectors $\mathbf{u_{n-k}} \cdots, \mathbf{u_n}$,
3. form matrix $U \in \mathbb{R}^{n \times k}$ with $\mathbf{u_{n-k}} \cdots, \mathbf{u_n}$ as columns, and matrix $Y = U^t$,
4. cluster the pointsy_i using the k-means into clusters $A_1, \cdots, A_k$,
5. build the communities $C_1, \cdots, C_k$ such $C_i = \{v_j | y_j \in A_i\}$.

For efficiency reasons it is preferable to use a normalized version of the Laplacian.
Shi and Malik [37] propose to use the following normalization

$$L_{\text{rw}} = D^{-1}L = I - D^{-1}W. \tag{32}$$

The notation L_{rw} is due to the fact that it is closely related to random walk. Ng, Jordan and Weiss [38] propose to use the following normalization

$$L_{\text{sym}} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2} \tag{33}$$

and furthermore impose to normalize U to have a norm equal to one. The notation L_{sym} is used because it is a symmetric matrix (see [39] for a complete review of the properties of the both normalizations). In practice it is preferable to use $L_{\text{rw}} = D^{-1}L$ because its eigenvectors are directly the cluster indicators $\mathbf{1}_{C_i}$, while the eigenvectors of L_{sym} are additionally multiplied by $D^{1/2}$, which might leads to undesirable effects. And moreover using L_{sym} also does not have any computational advantage [36].
Let us end on the complexity issue: spectral clustering requires the computation of the k eigenvectors of the Laplacian matrix, and if the graph is large, an exact computation require a time $O(n^3)$, which is penalizing for large graphs.

8 Galois Lattices

All the previous clustering methods are conceived to build non overlapping clusters, and are non exhaustive (i.e. one execution gives one solution), but there exists more informative methods. One of these interesting methods, is the Formal Concept Analysis (FCA) and the Galois Lattices [40] [41], which can be used for the conceptual clustering [42] [43]. A Galois Lattice clusters the data (called objects) in classes (called concepts) using their shared properties. The concepts found using Galois Lattices can be communities of people sharing connections, but also shared informations or opinions in an online network.
We introduce these lattices with a classical example, let us suppose that we have a description of some animal species (see table 1) : Lion, Finch, Eagle, Hare and Ostrich. This description is based of a list of properties such these species have or not: Ddo hey preying? Do they flying? Are they

	Preying	Flying	Bird	Mammal
Lion	x			x
Finch		x	x	
Eagle	x	x	x	
Hare				x
Ostrich			x	
Bee		x		
Bat		x		x

Table 1. A very simple incidence table description for a few animal species.

birds? Are they mammals? It is easy to compute a matrix of similarity counting the number of shared properties. As explained in section 4 we can turn these similarities in distances and then apply an ascendant hierarchical clustering. But in this case for each clustering one and only "point of view" will be shown. Figure 5 show a "multi-point of view" representation of the same informations, where species are labeled in red, while properties are labeled in blue. In this graph we see that Lions are the only ones (in our table) to be mammal and to prey, while Bats are the only ones to be mammal and to fly. In the same way, we see that Eagles are the only birds which both fly and prey, while it shares the two first properties with Finches. And these two flying birds, share this only property with Ostriches. We see in the graph that the concept "birds" is not the same that the concept "flying", because in the first one we have the "non-flying" Ostriches, and in the second one the "non-birds" Bats, which are mammals.
The reading rule of this kind of graph is the following: there is a downward path from a node v_i with a property p to a node v_j, only if v_j has also the property p. A consequence is the following: if there is two downward different paths from a node v_i with a property p to two different nodes v_j and v_k, then v_j and v_k share this property p.
This kind of graph is called a *Galois lattice.* The first use of lattices for observed structure data back to the attempt of formalize the logic of Quantum mechanics in [44]
We give now the formal definition of Galois lattices. Let us suppose that we have a set of objects O, and a set of possible attributes A for these objects. The possession of a property $a \in A$ by an object $o \in O$ is fulfilled when there is a relation I between them: aIo. Relations I between O and A are captured in a binary incidence matrix. The triplet $K = (O, A, I)$ is called a *formal context* or simply a context.
A *concept* is any couple $C = (X, Y) \subset O \times A$, which can forms a closed rectangle in the incidence matrix, i.e. we must have:

$$f(X) = \{a \in A | \forall o \in X, oIa\} = Y \tag{34}$$

$$g(Y) = \{o \in O | \forall a \in Y, oIa\} = X. \tag{35}$$

Functions f and g defined in (34) and (35) are called respectively the *intension* and the *extension* of the concept C. The intention is the set of the defining properties of the concept, while the extension is the set of the objects which forms the concept. The couple (f, g) is called a *Galois connection.*

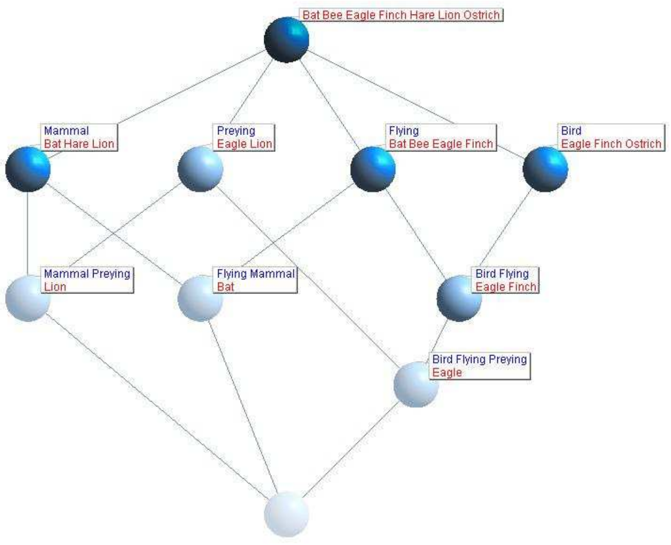

Fig. 5. Species Galois Lattice.

For the sake of illustration, in the table 1 the set $X = \{Finch, Eagle\}$ gives a concept, because $f(X) = \{Flying, Bird\} = Y$ and $g(Y) = X$, and this concept is $(\{Finch, Eagle\}, \{Flying, Bird\})$. On the contrary$X' = \{Lion, Hare\}$ do not gives a concept because $f(X') = \{Mammal\} = Y'$ and $g(Y') = \{Lion, Hare, Bat\}$, but this latter set gives the concept $(\{Lion, Hare, Bat\}, \{Mammal\})$.
Finally the *Galois lattice* is the set of concepts L with the following partial order $\leq$:

$$(X_1, Y_1) \leq (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2(\text{ or } Y_1 \supseteq Y_2). \tag{36}$$

The Galois lattice is denoted $T = (L, \leq)$ and its representation, as in figure 5, is called its *Hasse diagram.* As seen in the above example find a concept is not difficult:

1. pick a set of obects X,
2. compute $Y = f(X)$,
3. compute $X' = g(Y)$,
4. (X', Y) is a *formal concept* .

The dual approach can be taken starting with a set of attributes.
A more difficult task is, given a formal context table, to generate all the existing concepts and build the Hasse diagram of the Galois lattice. We detail here one of the simplest methods to build a Galois lattice, the Bordat's algorithm [45], which build L and its Hasse diagram.
Let us, firstly, define the cover of a concept $C = (X, Y)$, denoted $\underline{C}$

$$\underline{C} = \left\{C' | C' \leq C \text{ and } \nexists C'' : C' \leq C'' \leq C\right\}. \tag{37}$$

The algorithm starts with the set of concept given by the single objects $(o, f(o))$ and then find all their children nodes which are added to L and linked to their parent. This child generation process is iteratively repeated for every new pair:

```
– $L < -\{(o, f(o)) | o \in O\}$,
– for each concept $C \in L$:
    – build $\underline{C}$,
    – for each $C' \in \underline{C}$:
        • if $C' \notin L$, then $L < -L \cup \{C'\}$,
        • add the edge between $C$ and $C'$.
    – end for.
– end for.
```

Several other algorithms were proposed and the reader can take a look at [46] for a review and a comparison of performances.

The first use of Galois lattices in social network data analysis is owed to Freeman in [47], where he use the belonging to a clique as attribute. Another direct use of Galois lattices in the social network analysis is to use the adjacency matrix as incidence matrix. In the figure 6 we give an simple example of a graph containing three maximal cliques : $\{3, 5, 6\}$, $\{0, 2, 7\}$ and $\{0, 5, 6, 7\}$. In figure 7 we give the Galois lattice of this network, and even if it is look more complicated than the network, it is very easy to retrieve the cliques : it suffice to find concepts where both objects and attributes sets are equal. We surely could find concept like k-plex or k-core in the same way.

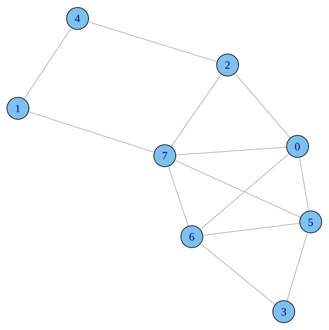

Fig. 6. A simple graph example.

Now, let us take a look to the complexity issue. If we denote $|O|$ the number of objects, $|A|$ the number of attributes and $|L|$ the size of the lattices (i.e. the number of concepts), then the different algorithms have a complexity time in $O(|O|^2|A||L|)$ or $O(|O||A|^2|L|)$ (see [46] for details).

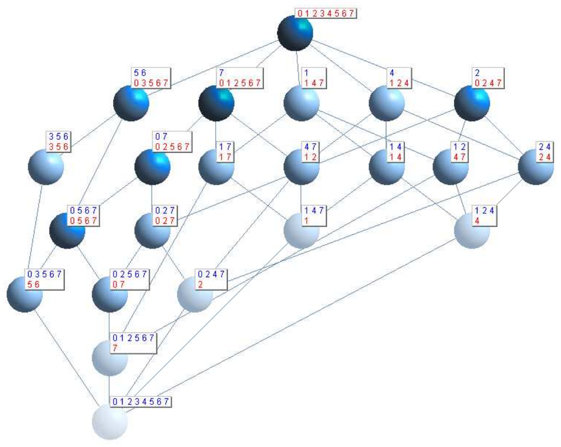

Fig. 7. The Galois Lattice of Social Network shown in the figure 6.

The Galois lattice based clustering is then costly, and moreover not simple to read but , but has several advantages in comparison with similarity clustering (see [48] for a good comparison). First, the notion of similarity is more strict in Galois lattices than in similarity based clustering: two objects in a Galois lattice are similar if they share identical properties, while they are alike to a degree quantified by a proximity measure in the other case. Secondly, given a dataset the Galois lattice turns a context into a unique and complete lattice of concepts, while classical clustering produces a result over all possible classes, depending of several choices (methods, parameters,...). Moreover, the resulting classes are, in most of the methods, disjoint whereas concepts in the lattice may overlap.

9 Discussion

As already pointed out forward the essential starting point in community detection, the definition of a community or cluster, is not an easy task [13]. Already in 1964, Bonner [49] argued that there could not be a universal definition of a cluster. In fact, what is a cluster is in the eyes of the beholder, and *one person's noise could be another person's signal* [3]. That is why we can write that *cluster analysis is structure seeking although its operation is structure imposing* [50].
Even with a given single and clear definition of a cluster, we expect from an adapted clustering that it gives a good clustering, but, again, what is a good clustering? Knowing that clustering algorithms transform clusters's research into optimization problem whose computational complexity is typically intractable and which are solved by approximation algorithms [13]. And in data clustering many choices must be done before any analysis (cluster definition , algorithms, measures, ...) which influence strongly the result. And the crucial choice of the algorithm will be influenced by

the sizes (number of vertices and/or number of edges) of the network, which has a direct impact on the time of execution.
But, in spite of all these warnings, clustering algorithms allow us to retrieve valuable pieces of information in social networks, by finding communities. But in the future, the community detection is not the only classification task in social networks analysis.

References

1. Freeman, L.: The Development of Social Network Analysis: A Study in the Sociology of Science. Empirical Press, Vancouver, BC, Canada (2004)
2. Wasserman, S., Faust, K.: Social Network Analysis. Cambridge Univ. Press (1994)
3. Han, J., Kamber, M.: Data mining: concepts and techniques. Morgan Kaufmann (2006)
4. Hand, D.J., Mannila, H., Smyth, P.: Principles of data mining. MIT Press (2001)
5. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**(2) (Feb 2004) 026113
6. Fortunato, S.: Community detection in graphs. Physics Reports **486**(3-5) (2010) 75–174
7. Shaeffer, S.E.: Graph clustering. Computer Science Review **1** (2007) 27–64
8. Moreno, J.L.: Emotions mapped by new geography (1933)
9. Cartwright, D., Harary, F.: Structural balance: A generalization of heider's theory. Psychological Review **63** (1956) 277–93
10. Scott, J.: Social Network Analysis. Sage (2000)
11. Zachary, W.W.: An information flow model for conflict and fission in small groups. Journal of Anthropological Research **33** (1977) 452–473
12. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA **99** (2002) 7821–7826
13. Estivill-Castro, V.: Why so many clustering algorithms: a position paper. SIGKDD Explor. Newsl. **4**(1) (2002) 65–75
14. Luce, R.D., Perry, A.D.: A method of matrix analysis of group structure. Psychometrika **14**(2) (1949) 95–116
15. Luce, R.D.: Connectivity and generalized cliques in sociometric group structure. Psychometrika **15** (1950) 169–190
16. Mokken, R.: Cliques, clubs and clans. Quantity and Quality **13** (1979) 161–173
17. Seidman, S., Foster, B.: A graph-theoretic generalization of the clique concept. Journal of Mathematical Sociology **6** (1978) 139–154
18. Seidman, S.B.: Internal cohesion of ls sets in graphs. Social Networks **5**(2) (1983) 97 – 107
19. Luccio, F., Sami, M.: On the decomposition of networks in minimally interconnected subnetworks. IEEE Transaction Circuit Theory **16**(2) (1969) 184 – 188

20. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proc. Natl. Acad. Sci. USA **101** (2004) 2658–2663
21. Bollobas, B.: Modern Graph Theory. Springer Verlag, New York, USA (1998)
22. Kannan, R., Vempala, S., Vetta, A.: On clusterings - good, bad and spectral. Journal of the ACM **51**(3) (2004) 497–515
23. Chung, F.R.K., Lu, L.: Complex graphs and networks. AMS (2006)
24. Freeman, L.: A set of measures of centrality based upon betweenness. Sociometry **40**(1) (1977) 35–41
25. Hartuv, E., Shamir, R.: A clustering algorithm based on graph connectivity. Inf. Process. Lett. **76**(4-6) (2000) 175–181
26. Burt, R.S.: Positions in networks. Social Forces **6** (1976) 93–122
27. Lorrain, F., White, H.: Structural equivalence of individuals in social networks. Journal of Mathematical Sociology **1** (1971) 49–80
28. MacQueen, J.: Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967). (1967)
29. Diday, E.e.a.: Optimisation en classification automatique. vol. i, ii. Technical report, Institut National de Recherche en Informatique et en Automatique (INRIA), Le Chesnay, France (1979)
30. Lance, G.., Williams, W.: A general theory of classificatory sorting strategies. i: Hierarchical systems. Comp. Journal **9** (1967) 373–380
31. Newman, M.E.J.: Detecting community structure in networks. The European Physical Journal B **38** (2004) 321–330
32. Brandes, U.: A faster algorithm for betweenness centrality. Journal of Mathematical Sociology **25** (2001) 163–177
33. Kernighan, B., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal **49**(2) (1970) 291308
34. Bui, T.N., Leighton, F.T., Chaudhuri, S., Sipser, M.: Graph bisection algorithms with good average case behavior. Combinatorica **7** (June 1987) 171–191
35. Sima, J., Schaeffer, S.: On the npcompleteness of some graph cluster measures. In Wiedermann, J., Tel, G., Pokorny, J., Bielikova, M., Stuller, J., eds.: Proceedings of the Thirtysecond International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM. Volume 383 of Lecture Notes in Computer Science., Berlin, Heidelberg, Germany, Springer Verlag GmbH (2006)
36. von Luxburg, U.: A tutorial on spectral clustering. Technical Report Technical Report 149, Max Planck Institute for Biological Cybernetics (2006)
37. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(8) (2000) 888–905
38. Ng, A., Jordan, M., Weiss, Y.: On spectral clustering: analysis and an algorithm. In: T. Dietterich and S. Becker and Z. Ghahramani (Eds), Advances in Neural Information Processing Systems. Volume 14. MIT Press (2002)

39. Chung, F.R.K.: Spectral Graph Theory. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society (1997)
40. Barbut, M., Monjardet, B.: Ordre et classification, Algebre et combinatoire, Tome 2. Hachette (1970)
41. Birkhoff, G.: Lattice Theory. Volume 25. American Mathematical Society, New York (1940)
42. Carpineto, C., Romano, G.: Galois: An order-theoretic approach to conceptual clustering. In: Proc. Of the 10th Conference on Machine Learning, Amherst, MA, Kaufmann. (1993) pp. 33–40
43. Wille, R.: Line diagrams of hierarchical concept systems. Int. Classif. **11** (1984) 77–86
44. Birkhoff, G., von Neumann, J.: The logic of quantum mechanics. Ann. Math. **37** (1936) 823–843
45. Bordat, J.: Calcul pratique du treillis de galois dune correspondance. Mathématique, Informatique et Sciences Humaines **24**(94) (1986) 31–47
46. Kuznetsov, S.O., Obedkov, S.A.: Comparing performance of algorithms for generating concept lattices. In: Concept Lattices-based Theory, Methods and Tools for Knowledge Discovery in Databases (CLKDD'01), Stanford, July 30, 2001. (2001)
47. Freeman, L.: Cliques, galois lattices, and the structure of human social groups. Social Networks **18** (1996) 173–187
48. Valtchev, P., Missaoui, R.: Similarity-based clustering versus galois lattice building: Strengths and weaknesses. In: Workshop Objects and Classification, A Natural Convergence. European Conference on Object-Oriented Programming. (2000)
49. Bonner, R.: On some clustering techniques. IBM Journal of Research and Development **8** (1964) 22–32
50. Aldenderfer, M.S., Blashfield, R.K.: Cluster Analysis. Sage Publication (1984)