# A Visual Data Mining Approach to Find Overlapping Communities in Networks

Jiyang Chen, Osmar R. Zaïane and Randy Goebel
Department of Computing Science
University of Alberta, Canada T6G 2E8
{jiyang, zaiane, goebel}@cs.ualberta.ca

## Abstract

*Communities in social networks may overlap, with some hub nodes belonging to multiple communities. They may also have outliers, which are nodes that belong to no community. The criterion to locate hubs or outliers is network dependent. Previous methods usually require this information as input parameters, e.g., an expected number of communities, with no intuition or assistance. Here we present a visual data mining approach, which first helps the user to make appropriate parameter selections by observing initial data visualizations, and then finds and extracts overlapping community structures from the network. Experimental results verify the scalability and accuracy of our approach on real network data and show its advantages over previous methods.*

## 1. Introduction

There has been a recent surge of research on finding communities in networks, which can be used to represent various kinds of complex systems in the real world. A community (or *cluster*) can be considered a subgraph such that the density of edges within is greater than the density of edges between its nodes and nodes outside [1]. Recent studies have also revealed that network models of many real world phenomena exhibit an overlapping community structure, i.e., a node can belong to more than one community. The participation of nodes in two or more communities is hard to manage with classical graph clustering methods where every vertex of the graph belongs to exactly one community [2]. This is especially true for social networks, where individuals can connect to several groups in the network as *hubs*. However, in real networks we also have another node category, which belongs to no community, i.e., *outliers*. Therefore, a typical social network consists of communities, hubs and outliers. It is essential for community discovery methods to identify nodes in these three categories, since the isolation of hubs and outliers can be crucial for many community-based applications.

Unfortunately, there doesn't yet exist a precise description of what a *community* really is. Moreover, the definition would naturally be different across domains, or even across different networks of the same domain. Therefore, most proposed approaches [1], [3], [4], [2], [5] require the user to describe the communities they are looking for by providing initial parameters, e.g., community size, density value, etc. However, appropriate parameters are usually extremely hard to determine without tedious and repeated testing.

In this paper, we describe ONDOCS (Ordering Nodes to Detect Overlapping Community Structure). Our visual data mining approach first generates visualizations of the network in question by ordering nodes based on their reachability scores; this helps the user understand the emerging network structure in order to choose appropriate parameters. After the initial visualization, selected parameters are used for extracting communities, hubs and outliers from the network. Similar visual data mining ideas are applied in [6], [7], [8] to help users determine parameters for decision tree construction, rule discovery, etc. Our work makes the following contributions:

- A visual data mining approach to assist the user in finding appropriate parameters to describe the communities they are looking for.
- A scalable and accurate method to discover communities, hubs, and outliers in social networks.

The rest of the paper is organized as follows. We discuss related work in Section 2. Section 3 presents the ONDOCS approach. We report experimental results in Section 4, followed by conclusions in Section 5.

## 2. Related Work

In general, there are two ways to detect overlapping communities in a network. One natural idea is to first globally partition the network and then locally expand the discovered communities to locate overlapping components. For example, for overlapping community discovery in a name-entity network, Li et al. [9] generate community cores by merging triangles (3-cliques) so that one vertex can be part of different communities if it belongs to several cliques. Similarly, Baumes et al. [10] initialize community cores using the Link Aggregate (LA) Algorithm and then refine the peripheries by an Iterative Scan (IS) procedure. Another mainstream research direction for this problem is based on fuzzy clustering. Zhang et al. [11] combine modularity and a fuzzy c-means clustering algorithm to identify overlapping communities. Nepusz et al. [4] propose a similarity function

based on membership, and solve the fuzzy community detection problem as a constrained optimization problem. Recently, Palla et al. [2] propose the CFinder system to partition complex networks to $k$-clique communities, where $k$ is a given as clique size. Gregory proposes the CONGA algorithm [1] based on the "betweenness" score [12] and later extends it to the CONGO algorithm to improve scalability [3]. He also shows that CONGO provides the same level of performance as CFinder, on synthetic networks.

While all these methods successfully detect overlapping communities, some major problems remain. Most methods do not consider outliers, thus many outliers would be classified into communities. These methods also intentionally focus on overlapping communities to the extent they find or force overlap even for data without such structure. More importantly, many approaches [1], [3], [9], [2], [11] require initial parameters that are not only difficult to determine but also highly sensitive.

## 3. Our ONDOCS Approach

### 3.1. Relationship Definition

Originally, ONDOCS is inspired by the OPTICS algorithm proposed by Ankerst et al. [13], where points are ordered for data clustering. However, unlike their clustering approach, we do not have a distance measure between nodes. Since the neighborhood around any two nodes in question is important in assessing their relationship in social networks, we define the relationship $R$ between node $i$ and $j$ as follows:

$$R(i,j) = \frac{\sum_{x \in N_j} r(i,x) + \sum_{x \in N_i} r(x,j)}{2} \qquad (1)$$

where $N_i$ is the neighbourhood of node $i$, including $i$ itself and all nodes that connect to $i$. The similarity between node $i$ and $j$ is defined as the average of $R(i \rightarrow j)$, representing the relations from $i$ to $j$'s neighbourhood, and $R(j \rightarrow i)$, representing relations from $j$ to $i$'s neighbourhood. $R(i \rightarrow j)$ is defined as the sum of relation scores between $i$ and all nodes in $j$'s neighbourhood, similarly for $R(j \rightarrow i)$ with respect to $j$ and $i$'s neighbourhood. Next, in order to quantify the relationship $r(i,j)$ between node $i$ and $j$, we compare the probability of the event that $i$ *and* $j$ *are connected* in the original graph $G$, to a random model $G'$, where we keep only the same node number $n$ and node degree $k_1, ..., k_n$ and leave the rest random. Only if the probability of having two nodes connected in the random model is low, does the fact that they are indeed connected show us a strong relationship. In $G'$, it is obvious that the probability of node $i$ having a connection to any other node is $P(i) = \frac{k_i}{n-1}$ (similarly, $P(j) = \frac{k_j}{n-1}$). Here we assume $G'$ is undirected so that the event of $i$ connecting to $j$ and $j$ connecting to $i$ is equivalent, thus the probability of $i$ and $j$ being connected is the maximum of $P(i)$ and $P(j)$. In other words, with respect to $i$, the probability of selecting $j$ as one of $i$'s neighbours is $\frac{k_i}{n-1}$. We cannot achieve a higher score unless $k_j > k_i$, thus the probability of the fact that two nodes are connected in our model is decided by the node with the higher degree. Note that $P(i \leftrightarrow j) \neq P(i) * P(j)$ since the two events $i$ connecting to $j$ and $j$ connecting to $i$ are dependent on each other. Therefore we have

$$P(i \leftrightarrow j) = max(P(i), P(j)) = \frac{max(k_i, k_j)}{n-1} \qquad (2)$$

We define the relation score $r(i,j)$:

$$r(i,j) = A_{ij} - \frac{max(k_i, k_j)}{n-1} \qquad (3)$$

where $A_{ij} = 1$ if $i$ and $j$ are connected, 0 otherwise. The generalization for directed or weighted graphs is straightforward.

### 3.2. Ordering Nodes to Visualize Networks

Now we generate network visualizations by ordering nodes based on their relation scores. Given the relationship function $R$, for node $n_i$, we create a list of nodes $l_i$ ordered by their relation to $n_i$ from high to low. (Note that we can limit candidate nodes to those which have $R > 0$, i.e., they are connected to or share at least one neighbour with $n_i$.) We define the $k^{th}$ value in this list to be $l_{ik}$. Here, our approach takes one input parameter $s$. However, as we will show in Section 4, $s$ does not strongly affect the output. In practice, we usually generate several visualizations, with $s$ ranging from 2 to 8, and let the user select an $s$ based on their observations. For a node $n_i$, we define its community score $C_s$ to be the $s^{th}$ value in its node list $l_i$, i.e., $C_s(n_i) = l_{is}$, and $C_s(n_i) = 0$ if there are less than $s$ nodes in the list. We define the reachability of node $j$ with respect to $i$ as

$$reach_s(i,j) = \begin{cases} R(i,j) & if\ C_s(n_i) > R(i,j) \\ C_s(n_i) & otherwise \end{cases}$$

Intuitively, the parameter $s$ represents the expected number of nodes that one node is similar with in order to be a member of any community. $C_s$ is the lowest relation score between node $i$ and its similar neighbours in one community. In this way, $reach_s(i,j)$ measures the community relationship between $i$ and $j$. It is their direct distance score if $i$ and $j$ are far away from each other, and is equal to the community radius of $i$ if $j$ is close enough. Therefore, a decreasing order of the reachability scores ($RS$) indicates a node list for $i$, starting from $i$'s most related neighbours to the least ones.

We present our algorithm for generating node lists ordered by their RS scores as Algorithm 1. More specifically, our algorithm creates an ordering of network nodes, additionally storing a reachability score $RS(i)$ for each node $i$. It starts at a given node $n_{start}$ and inserts $n_{start}$ into a max-heap

**Algorithm 1** The ONDOCS Algorithm

**Input:** A social network $G$ with $n$ nodes and $m$ edges, a start node $n_{start}$ and possible $s$ values $s_0, s_1, s_2....$

**Output:** A list of nodes $L$ with their Reachability Scores $RS$ for each $s$.

**1.** Sort a node list $l_i$ for each node $n_i$, ordered by their relation score to $n_i$, from high to low.

**2.** For each $s$ :

Initialize a max-heap $h$, insert $n_{start}$ in $h$ with $RS = 0$.
Select the $s^{th}$ largest element in $l_i$ for each node $n_i$ as its community score $C_s(n_i)$.
While (there is still nodes in heap $h$) :
Pop the node $\alpha$ in $h$ with largest value $\epsilon$.
Store $\alpha$ in $L_s$ with $RS_\alpha = \epsilon$.
For all nodes $x$ in $l_\alpha$:
If $x \notin h$, insert $x$ into $h$ with $reach_s(\alpha, x)$.
If $x \in h$, update its value if $reach_s(\alpha, x)$ is larger.
Update max-heap $h$.

**3.** Return list $L_s$ with $RS$ values for each $s$ value.

structure $h$, which is maintained to store the reachability of candidate nodes. At each step, the node $j$, which has the highest reachability score in $h$, is chosen to be the next node in order and the popped score is stored as $RS(j)$. All nodes that are in $j$'s neighbourhood are then inserted into $h$ with their reachability according to $j$, if they are not yet in $h$. The value in $h$ is updated if the node is already in $h$ and its new score is higher. Then $h$ is updated to maintain its max-heap property. The algorithm stops after all nodes in the network are visited and produces a sequence of nodes with their reachability scores for each $s$ value, which can be visualized as a 2D graph by GNUplot [14] (See Figure 2).

The overall complexity of ONDOCS is $O(n \log n)$. The list generation and sort step takes $O(cn)$ where constant $c$ is the average number of similar nodes for each node. Note that, based on our relationship function, one node can only be similar to another if they are connected or share one or more neighbours. In step 2, there are $n$ insertions to the heap $h$ and updating $h$ for each insertion takes $O(\log n)$ time in the worst case. However, as shown in Section 4, the actual running time of our algorithm is close to $O(n)$.

### 3.3. Extracting Overlapping Communities

We have generated lists of nodes given specific $s$ values, where we found that the ordering of the corresponding $RS$ values has interesting community properties. For example, if we start from one node $i$, we will first visit other nodes in $i$'s community in sequence. This is because the reachability score from $i$ to these nodes are always higher than nodes outside $i$'s community. Therefore, each community can be seen as a group of consecutive nodes with high $RS$ scores. In other words, each "mountain" in the 2D visualization represents a community. A noticeable drop of subsequent $RS$ scores after a "mountain" indicates that this community has ended. The "valley" between two "mountains" represents a collection of hubs, which belong to several communities (See Figure 3). For instance, if we start from nodes in community $\alpha$, the fact that hubs have neighbours from different communities makes $RS$ scores of hubs lower than that of those single-community nodes in $\alpha$ but still higher than nodes in communities other than $\alpha$. Therefore, after all single-community nodes in $\alpha$ are visited, hubs are next to follow before nodes in other communities; these form the "valley" between "mountains."

As we have discussed in the introduction, there is no global community definition, thus communities in specific networks need to be defined by parameters given by the user. While parameters for previous methods are hard to determine, our visual data mining approach generates visualizations with different $s$ values first. After the user chooses the suitable one based on their observations, they need to further provide two parameters to define the communities in this network, *Community Threshold (CT)* and *Outlier Threshold (OT)*. From the first node as the starting community, we scan all nodes along the list. One node $n_i$ is merged into the current community if $RS(n_i) \geq CT$. If $CT > RS(n_i) > OT$, $n_i$ is classified as a hub. If $OT \geq RS(n_i)$, it is an outlier. Since the first node of a community in the list always has low RS scores, e.g., the starting node always has $RS = 0$, we refine the outlier and hub nodes by moving any node $n_i$ into corresponding communities if we have $RS(n_{i+1}) \geq CT$.

To represent the idea that hubs can belong to $k$ communities, for each hub node $i$, we use a vector of "belonging factors" $v = (f_{(i,1)},\ f_{(i,2)}\ ...\ f_{(i,k)})$ where each coefficient $f_{(i,k)}$ measures the strength of the relationship between node $i$ and community $k$. For every community $C_k$, we can quantify the Overall Relationship between $i$ and $C_k$ as

$$OR_{(i,k)} = \begin{cases} \sum_{x \in C_k} R(i,x) & if\ \sum_{x \in C_k} R(i,x) > 0 \\ 0 & otherwise \end{cases}$$

We then normalize the vector to get the coefficients so that we have $\sum_{x=1}^{k} f_{(i,x)} = 1$. Therefore, one node can belong to many communities at the same time, weighted by the relationship value in the range $[0, 1]$ and the sum of belonging coefficients to communities is the same for all nodes in the network, except outliers.

In summary, our approach to the community mining process is aided by visual data mining. Instead of asking the user to arbitrarily provide vital initial parameters, we generate network visualizations so that the user can observe the emerging community structure before appropriate parameters can be determined. While parameters can be easily altered, the impact on the change can be clearly visualized.

| Datasets | Vertices | Edges | Runtime / s | | | |
|---|---|---|---|---|---|---|
| | | | CONGO [3] | | CF [2] | ONDOCS |
| | | | h = 3 | h= 2 | | |
| football [5] | 180 | 787 | 8 | 2 | 1 | $< 1$ |
| protein_protein [2] | 2640 | 6600 | 114 | 11 | 3 | 11 |
| blogs [3] | 3982 | 6803 | 41 | 8 | 4 | 12 |
| PGP [15] | 10680 | 24316 | 772 | 104 | >20000 | 62 |
| word_association [2] | 7207 | 31784 | 15922 | 230 | 102 | 161 |
| blogs2 [3] | 30557 | 82301 | 15148 | 380 | 319 | 269 |
| cond-mat [16] | 27519 | 116181 | $> 20000$ | 1486 | 490 | 544 |

Table 1. Results on Real World Networks

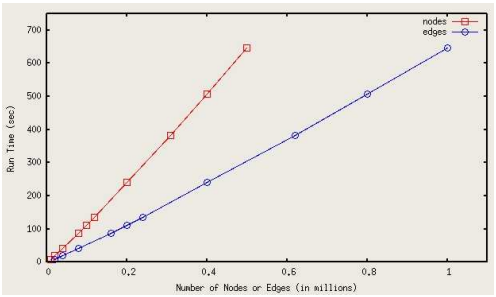


Figure 1. Algorithm Running Time

## 4. Experiment Results

Here we evaluate the ONDOCS approach using both synthetic and real world datasets. The performance of ONDOCS is compared with CFinder [2] and CONGO [3], which are shown to be two of the most efficient algorithms for finding overlapping communities [3]. The comparison is measured by the well known F-measure score and Adjusted Rand Index (ARI) [17]. All experiments were conducted on a PC with a 3.0 GHz Xeon processor and 4GB of RAM.

### 4.1. Scalability

To evaluate the scalability of our algorithm, we generated ten random graphs of vertices ranging from 10,000 to 500,000 and the number of edges ranging from 20,000 to 1,000,000. The edges are randomly distributed in the network. Figure 1 shows the performance of our algorithm on those networks. It clearly illustrates that although the running time of ONDOCS is $O(n \log n)$, in the worst case, our approach actually runs very close to linear time with respect to the number of vertices and edges.

To further evaluate the efficiency of the algorithm, we apply all three algorithms on several real-world networks. Table 1 shows the source of each network, its statistics, and the execution times for CONGO to compute the entire dendrogram, CFinder (v1.21) to generate solutions for $3 \leq k \leq 8$ and ONDOCS to create dataset visualizations for $2 \leq s \leq 8$. From the table, we can see that ONDOCS works well overall, while CONGO's running time increases dramatically with respect to $h$ and CF's clique detection becomes slow on some specific networks. However, for lack of ground truth with these datasets, to validate the accuracy of our results we use other real world datasets for which we have ground truth.

### 4.2. Accuracy

The first dataset we examine is the schedule for 787 games of the 2006 National Collegiate Athletic Association (NCAA) Football Bowl Subdivision (also known as Division 1-A) [5]. In the NCAA network, there are 115 universities divided into 11 conferences. In addition, there are four independent schools, namely Navy, Army, Notre Dame and Temple, at this level, and 61 schools from lower divisions. Each school in the division plays more often with schools in the same conference than schools outside. Independent schools do not belong to any conference and can play with teams in all conferences, while lower division teams only play very few games. In other words, this network contains 180 vertices (115 nodes as 11 communities, 4 hubs and 61 outliers), connected by 787 edges.

First, the ONDOCS approach generates several visualizations with different $s$ values for the user to choose. We show three of them in Figure 2. As we can see, the images are very similar. The larger the $s$ value is, the smoother the curves are and the fewer "spikes" we have. Nevertheless, all three visualizations clearly represent the network structure, where there are 11 communities, a few hubs and a bunch of outliers.

The selection of parameters is based solely on users' visual interpretation of the visualized network. First we choose the visualization with $s = 2$, where the community structure is shown in most detail since pair relations are mostly measured as direct distance. In Figure 2, we note that nodes in sequence from 120 to 180 are barely related to the rest and can be considered as outliers, therefore we

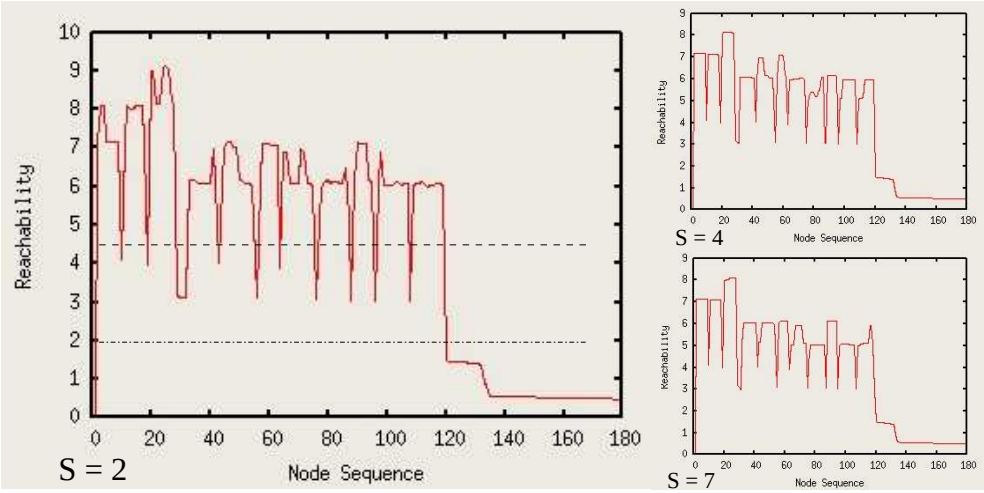


Figure 2. Visualization for Football Dataset

| Algorithm | 115 Nodes in 11 Clusters | | | Plus 4 Hubs | | | Plus 4 Hubs and 61 Outliers | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | cluster | Hub | ARI | cluster | Hub | Fm# | cluster | Hub | Fm# | Outliers | Fm# |
| CONGO (h=2) | 11* | 92 | 0.047 | 11* | 100 | 0.038 | 11* | 96 | 0.04 | 0 | 0 |
| CF (k=4) | 11 | 6 | 0.945 | 12 | 8 | 0.167 | 12 | 8 | 0.167 | 61 | 1.00 |
| ONDOCS (s=2) (CT = 4.5, OT = 2) | 11 | 0 | 1.00 | 11 | 3 | 0.857 | 11 | 3 | 0.857 | 61 | 1.00 |

Table 2. Result Comparison on the Football Dataset. (*The right cluster number is provided as a parameter for the CONGO algorithm.) (# Fm represents F-measure score.)

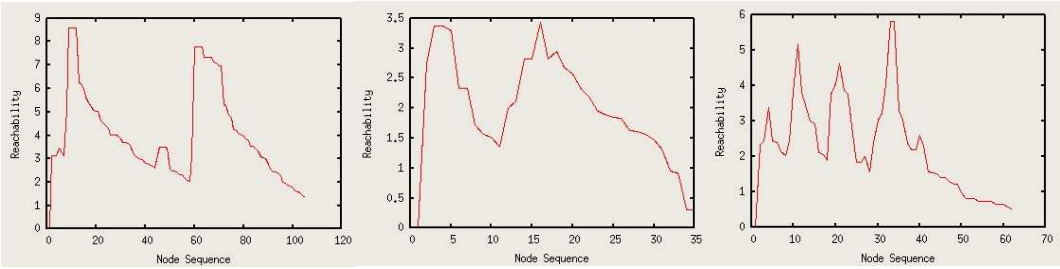


Figure 3. Visualization for Other Datasets

set $OT = 2$. Furthermore, we see a community usually ends with a $RS$ score between 3 and 5, thus we set $CT = 4.5$ so that all communities are separated. The thresholds are shown as disconnected lines in the figure.

To evaluate how these algorithms detect overlapping communities, we provide the data to in three different ways. At first, we only give 115 community nodes and connections between them, then we measure the accuracy of output communities by the ARI score based on the ground truth, which is the conference assignment. Then we add the 4 hubs and their connections into the network. Although these hubs clearly belong to multiple communities, we do not have exact ground truth for which communities these hubs should go. Therefore, we measure the accuracy of the output hubs by the F-measure score, which is defined as the harmonic mean of precision and recall. Finally we give the complete network with communities, hubs and outliers. Table 2 shows the experimental results for three algorithms. As we can see, the CONGO algorithm always detects overlaps, even for the first network where there are only community nodes. Additionally, it requires the cluster number as the input parameter, which is usually unavailable for real world networks, and it

still fails to find any outliers. The CF algorithm gives its best result when $k = 4$, where it detects all outliers and finds 12 clusters, which is very close to the truth. However, CF also finds hubs when there is no overlap and the accuracy of its overlap detection is low with only a 0.167 F-measure score. Our ONDOCS algorithm works the best overall. It finds all outliers and only detects hubs when there is indeed some overlap between communities. The hub detection accuracy is not perfect, however, when we look more closely at the data, we find that the only missing hub team (Temple) plays half of its games (6 out of 12) with teams from the Mid-American conference, which explains why it is classified into that community.

We also apply our algorithm on other real world networks, including Political Book network [18], Mexican Politician network [19] and Dolphin network [12]. Although we do not have overlapping truth for these networks, approximate community information is provided by previous research: The political books can be categorized into two big communities, "liberal," "conservative" and one small community "neutral" in between; The Mexican politicians can be classified based on their backgrounds as "citizen" or "military"; Finally, the dolphins can be divided into four main groups. Due to lack of space, we only show visualizations for these datasets generated by ONDOCS in Figure 3. One can see that the images correctly depict the approximate community information we have. Accurate $CT$ and $OT$ values should be easy to determine by observation.

## 5. Conclusions

This paper proposes a visual data mining approach to detect overlapping communities in networks. Our method first generates lists of nodes, ordered by their reachability scores. Network visualizations are then provided to help the user incrementally determine important parameters. Finally, overlapping communities, i.e., communities, hubs and outliers, are extracted based on these parameters. Experimental results show that our approach not only scales well for large networks, but also achieves high accuracy for real world networks. Unlike previous approaches, our method only detects overlap when overlap exists. Moreover, appropriate parameters are easy to obtain by means of visual data mining.

## 6. Acknowledgments

Our work is supported by the Canadian Natural Sciences and Engineering Research Council (NSERC), by the Alberta Ingenuity Centre for Machine Learning (AICML), and by the Alberta Informatics Circle of Research Excellence (iCORE).

## References

[1] S. Gregory, "An algorithm to find overlapping community structure in networks," in *PKDD*, 2007, pp. 91–102.

[2] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, p. 814, 2005.

[3] S. Gregory, "A fast algorithm to find overlapping communities in networks," in *PKDD*, 2008, pp. 408–423.

[4] T. Nepusz, A. Petróczi, L. Negyessy, and F. Bazso; "Fuzzy communities and the concept of bridgeness in complex networks," *Physical Review E*, vol. 77, 016107, 2008.

[5] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "Scan: a structural clustering algorithm for networks," in *KDD*, 2007, pp. 824–833.

[6] M. Ankerst, C. Elsen, M. Ester, and H.-P. Kriegel, "Visual classification: an interactive approach to decision tree construction," in *KDD*, 1999, pp. 392–396.

[7] J. Han and N. Cercone, "Ruleviz: a model for visualizing knowledge discovery process," in *KDD*, 2000, pp. 244–253.

[8] S. T. Teoh and K.-L. Ma, "Paintingclass: interactive construction, visualization and exploration of decision trees," in *KDD*, 2003, pp. 667–672.

[9] X. Li, B. Liu, and P. S. Yu, "Discovering overlapping communities of named entities," in *PKDD*, 2006, pp. 593–600.

[10] J. Baumes, M. K. Goldberg, and M. Magdon-Ismail, "Efficient identification of overlapping communities," in *ISI*, 2005, pp. 27–36.

[11] S. Zhang, R. Wang, and X. Zhang, "Identification of overlapping community structure in complex networks using fuzzy c-means clustering," *Physica A*, vol. 374, pp. 483–490, 2007.

[12] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, 2004.

[13] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *SIGMOD*, 1999, pp. 49–60.

[14] Gnuplot, "http://www.gnuplot.info/."

[15] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas, "Models of social networks based on social distance attachment," *Phys. Rev. E*, vol. 70, no. 5, p. 056122, 2004.

[16] M. E. J. Newman, "The structure of scientific collaboration networks," in *PNAS USA, 98:404-409*, 2001.

[17] K. Y. Yip and M. K. Ng, "Harp: A practical projected clustering algorithm," *IEEE TKDE*, vol. 16, no. 11, pp. 1387–1397, 2004.

[18] V. Krebs, "http://www.orgnet.com/."

[19] Pajek, "http://vlado.fmf.uni-lj.si/pub/networks/pajek/."