Acta Cybernetica **20** (2011) 69–85.

Community Detection by using the Extended Modularity

Erika Griechisch and András Pluhár

Abstract

This article is about community detection algorithms in graphs. First a new method will be introduced, which is based on an extension [16] of the commonly used modularity [17, 18, 19, 20] and gives overlapping communities. We list and compare the results given by our new method and some other algorithms yileding either overlapping or non-overlapping communities. While the main use of the proposed algorithm is benchmarking, we also consider the possibility of hot starts, and some further extensions that considers the degree distribution of the graphs.

Keywords: graph mining, community detection, fuzzy partition, modularity

1 Motivation

Graph mining has became an important field of data mining, especially the community detection is very popular, because it can be applied in sociology, computer science, biology or finance.

The main problem in community detection is that there is no exact definition for communities; all we can do is to postulate some properties. One approach is that a community is a set of vertices that are relatively densely connected to each other, but sparsely connected to other dense groups in the graph. Several methods are proposed for community detection. The majority of those define non-overlapping communities (clusters, partitions) [24, 13, 2], some of them define overlapping sets (henceforward *communities*), see [1, 22, 11].

However, it became clear that overlapping communities must be also considered in Small World graphs, see [10, 12, 22, 16]. Moreover, the given algorithms turned out to be useful in data mining and modeling [8, 9]. In fact, one way to evaluate the performance of different algorithms is to check their predictive value in models [11]. Another possibility is to define plausible measures. The most successful of those is the Newman modularity that is defined for a clustering of a graph. The higher the modularity, the better the clustering [17]. Nevertheless, for large graphs the modularity is just approximated [7, 5], since it is an NP-hard problem to maximize modularity [6].

DOI: 10.14232/actacyb.20.1.2011.6

Népusz et al. [16] proposed an algorithm for the detection of overlapping communities in quadratic programming terms. To measure their results they also extended the notion of modularity to overlapping communities. We continue their work, but change the objective function, and examine the possibility of directly optimizing the extended modularity. Of course, one cannot expect an effective algorithm for this, since it is also a quadratic optimization problem. Still, it can be computed for small graphs, and provides good benchmarks for the community detection heuristics. It is also instructive, how good approximation of the modularity is achieved by those heuristics?

Other methods for extending modularity exist. Nicosia et al. [21] introduce a general extension of the modularity on directed graphs which moves from simple considerations about the meaning and structure of the original modularity function, using an enriched null–model.

2 Definitions

A graph G with vertex set V and edge set E will be denoted $G = (V, E)$, the cardinality of the vertex set will be denoted n, and the cardinality of the edge set will be denoted m. In this paper every graph is undirected and unweighted. The objective of classical community detection in networks is to partition the vertex set of the graph into c distinct subsets in a way that puts densely connected groups of vertices in the same community. Here c can either be given in advance or determined by the community detection algorithm itself. If c is known, a convenient representation of a given partition is the partition matrix $\mathbf{U} = [u_{ik}]$. $\mathbf{U}$ has $n = |V|$ rows and c columns, and $u_{ik} = 1$ if and only if vertex i belongs to the kth subset in the partition, otherwise it is zero [3]. From the definition of the partition, it follows that $\sum_{k=1}^{c} u_{ik} = 1$ for all $1 \leq i \leq n$. The size of community k can then be calculated as $\sum_{i=1}^{n} u_{ik}$, and for any meaningful partition, we can assume that $0 < \sum_{i=1}^{n} u_{ik} < n$. These partitions are called *hard* or *crisp* partitions, because a vertex can belong to one and only one of the detected communities [3]. The generalization of the hard partition follows by allowing u_{ik} to attain any real value from the interval $[0, 1]$. The constraints imposed on the partition matrix remain the same:

$$u_{ik} \in [0, 1], \qquad \forall 1 \leq i \leq n, 1 \leq k \leq c; \tag{1}$$

$$\sum_{k=1}^{c} u_{ik} = 1, \qquad \forall 1 \leq i \leq n; \tag{2}$$

$$0 < \sum_{i=1}^{n} u_{ik} < n, \qquad \forall 1 \leq k \leq c. \tag{3}$$

Equation 2 simply states that the total membership degree for each vertex must be equal to one. Informally, this means that vertices have a total membership degree of one, which will be distributed among the communities. Inequality 3 is the formal

description of a simple requirement: we are not interested in empty communities (to which no vertex belongs to any extent), and we do not want all vertices to be grouped into a single community. Partitions of this type are called *fuzzy partitions*.

Several methods have been developed to search fuzzy clusters (see e.g. [4]), but we can not apply these to graph partitioning, because these methods need some additional data.

In [16] Népusz et al. use a different approach using vertex similarities. They observe that a meaningful partition should group vertices that are somehow similar to each other in the same community and assume that a similarity function $s(\mathbf{U}, i, j)$ satisfies the following criteria:

- $s(\mathbf{U}, i, j) \in [0, 1]$
- $s(\mathbf{U}, i, j)$ is continuous and differentiable for all u_{ij}
- $s(\mathbf{U}, i, j) = 1$ if the membership values of vertex i and j suggest that these are as similar as possible
- $s(\mathbf{U}, i, j) = 0$ if the membership values of vertex i and j suggest that these are completely dissimilar.

From now on, we denote $s(\mathbf{U}, i, j)$ by s_{ij}. We use the similarity matrix from [16], that is

$$s_{ij} = \sum_{k=1}^{c} u_{ik} u_{jk}. \tag{4}$$

In [16] the following objective function was optimized

$$D_G(\mathbf{U}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (\tilde{s}_{ij} - s_{ij})^2, \tag{5}$$

where w_{ij}'s are optional weights and $\tilde{s}_{ij}$ is a prior assumption of the actual similarity of the vertices. Instead of using Equation 5, we extend the commonly used modularity [17, 18, 19, 20], and optimize this objective function directly over the space of feasible solutions.

Modularity is a property of a network and a specific proposed division of that network into communities. The measure of a division is large, if most of the edges are within clusters and only a few are between those.

Let A be the adjacency matrix of the network, k_i the degree of vertex i, $\delta(c_i, c_j)$ is the Kronecker delta and suppose that vertex i belongs to community c_i. Then

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

is the modularity of the given division.

The modularity can be either positive or negative. Positive values indicate the possible presence of community structure. Thus one can search for community

structure precisely by looking for the divisions of a network that have positive, and preferably large values of the modularity.

By replacing $\delta(c_i, c_j)$ with s_{ij} we get a quadratic function called extended (fuzzy) modularity [16]:

$$Q' = \frac{1}{2m}\sum_{i,j}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j) = \frac{1}{2m}\sum_{i,j}\left[A_{ij} - \frac{k_i k_j}{2m}\right]s_{ij} \quad (6)$$

$$= \frac{1}{2m}\sum_{i,j}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\langle u^i, u^j\rangle, \quad (7)$$

where u^i is the ith row in the fuzzy partition matrix $\mathbf{U}$ and $\langle ., . \rangle$ denotes the dot product.

So our the problem is to

$$\begin{array}{lll} \text{maximize} & Q' & \\ \text{subject to} & u_{ik} \in [0,1]\,, & \forall 1 \le i \le n, 1 \le k \le c; \\ & \sum_{k=1}^{c} u_{ik} = 1, & \forall 1 \le i \le n; \\ & 0 < \sum_{i=1}^{n} u_{ik} < n, & \forall 1 \le k \le c. \end{array}$$

Higher modularity usually means better division, so our aim was to maximize the extended modularity directly, and thereby gaining a benchmark compare the results given by other methods.

3 Examined methods

This section introduces the compared methods. The results of these methods were not just compared in a sense of modularity value, but the detected communities were given to the quadratic solver as an initial point, thus we used their results as a "hot starts."

In the case of overlapping communities, it is not obvious how to choose the community membership vectors. We decided to use uniform distribution, e.g. if the CPM (or N^{++}) method produces 4 communities and the community 1 and 2 also contains vertex v, then the membership vector of v is $(0.5, 0.5, 0, 0)$.

Table 1 shows the running time of the examined methods depending on the number of nodes n, the number of edges m and the average degree d_{avg}.

3.1 Community detection based on edge betweenness

The method of Girvan and Newman [18] is based on the definition of edge betweenness. The edge betweenness of an edge is the number of shortest paths between pairs of vertices run along it. Iteratively removing the edges with highest betweenness, we can determine a hierarchical tree and then communities.

3.2 Leading eigenvector method

Newman showed in [19] that the modularity can be expressed in terms of the eigenvectors of a characteristic matrix for the network, which he calls the *modularity matrix*, and that this expression leads to a spectral algorithm for community detection that returns better results in shorter running times than competing methods, e.g. edge betweenness.

3.3 Greedy modularity optimization method

Clauset, Newman and Moore presented a hierarchical agglomeration algorithm for detecting community structure [20]. This algorithm uses a greedy optimization in which, starting with each vertex being the sole member of a community of one, repeatedly join together the two communities whose amalgamation produces the largest increase in the modularity Q.

3.4 Label propagation

The label propagation method [25] is based on the following iteration. Suppose that a node x has neighbors and each neighbor carries a label denoting the community to which they belong to. Then x determines its community based on the labels of its neighbors. We assume that each node in the network chooses to join the community to which the maximum number of its neighbors belong to, such that the occurring ties are broken uniformly randomly.

At the beginning every vertex gets a unique label. Iteratively at every step, each node updates its label based on the labels of its neighbors. The iterative process continues until no node in the network changes its label.[1]

3.5 Walktrap

The walktrap method uses the idea of Markov chains, and it is based on random walks to tell a distance for every pair of vertices in the graph and with this it generalizes the distance of clusters [23].

3.6 Clique percolation

The clique percolation method (CPM) uses adjacent cliques to determine overlapping communities [12, 22]. It builds up the communities from k-clique (fully connected graphs), and two k-cliques are considered adjacent if they share $k-1$ nodes. Then communities are the unions of the adjacent k-cliques. Note that the result heavily depends on the value of the parameter k.

[1] The algorithm may give oscillation if the input is a bipartite graph; this case should be handled.

3.7 N^{++} method

The

$$N^{++}(v) = N^{+}(N^{+}(v)) = \{u \in V \mid d(u,v) \leq 2\}$$

is the set of vertices which are "close" to v. Using these sets and their dense subsets, the method provides overlapping communities [10, 11].

Table 1 shows the running time, where n is the number of vertices, m is the number of edges in the graph.

Table 1: Running time of the compared methods

Method	**Running time** worst case	
Edge betweenness	$\mathcal{O}(m^2 n)$	[18]
Leading eigenvector	$\mathcal{O}(m + n^2 \cdot steps)$	[19]
Newman greedy	$\mathcal{O}(m d_{\text{avg}} \log n)$	[20]
Label propagation	$\mathcal{O}(m+n)$	[25]
Walktrap	$\mathcal{O}(mn^2)$	[23]
CPM	$\mathcal{O}(\exp(n))$	[12]
N^{++}	$\mathcal{O}(\exp(n))$	[10]

3.8 Sequential greedy method

We also proposed a new sequential greedy (SG) method that might be considered as a heuristic for the modularity optimization problem. We determined the number of the communities (c) as the size of a maximal independent set I and placed those into different clusters. (The membership vector of the kth vertex of I is e_k, the the kth element of the standard orthonormal base.) Then we greedily decide about the membership vector of the leftover vertices using a partial modularity based on the already placed vertices. Since the extended modularity Q' (see Equation 7)

$$Q' \quad = \quad \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \langle u^i, u^j \rangle,$$

in step j we set u^j such that it maximizes the partial modularity function Q_j, where

$$Q_j = Q(u^j) = \frac{1}{2m} \sum_{i<j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \langle u^i, u^j \rangle = \frac{1}{2m} \left\langle \sum_{i<j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] u^i, u^j \right\rangle,$$

and u^j satisfies the constraints 2, 3. Obviously the function $Q(u^j)$ reaches its maximum if u^j equals the kth element of the standard orthonormal base, where k is the position of the largest coordinate of the vector $\sum_{i<j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] u^i$.

4 Technical background

This section describes the main software tools which were used during the research.

4.1 igraph library

The *igraph* [29] is a free software package for creating and manipulating undirected and directed graphs. It includes implementations for classic graph theory problems like minimum spanning trees and network flow, and also implements algorithms for some recent network analysis methods, such as the community structure search.

The igraph contains functions for generating regular and random graphs, manipulating graphs, assigning attributes to vertices and edges. It can calculate various structural properties, graph isomorphism, includes heuristics for community structure detection and supports many file formats.

We used the igraph functions `IC_edge_betweenness`, `IC_leading_eigenvector`, `IC_fastgreedy`, `IC_spinglass` and the `IC_walktrap` functions (where `IC` equals to `igraph_community`) to determine the corresponding community structure and the function `igraph_modularity` to determine the modularity value of the clustering.

4.2 octave

GNU Octave is a high-level language, primarily intended for numerical computations which is mostly compatible with Matlab. We used the octave function `sqp` to optimize the objective function which is an implementation of the SQP.

The sequential quadratic programming (SQP) is one of the most popular and robust algorithms for nonlinear continuous optimization. The method is based on solving a series of sub-problems designed to minimize a quadratic model of the objective subject to a linearization of the constraints.

4.3 Visualize graphs

Several methods exist for visualizing graphs, we used the `igraph_layout_graphopt` function, which optimizes vertex layout via the graphopt algorithm.

In contrast to other graph optimizers, graphopt does not use a finite-pass approach to layout optimization. Instead, it uses basic principles of physics to iteratively determine an optimal layout.

Each node is given a mass and an electric charge, and each edge is represented as a spring. The node mass, the electric charge, the optimal spring length, and the spring constant are tweakable in the gui in real time [30].

4.4 Communities

We used CFinder for CPM and the implementation of the N^{++} method is due to Csizmadia [1, 10]. We would like to express our thanks for obtaining free access to the appropriate softwares.

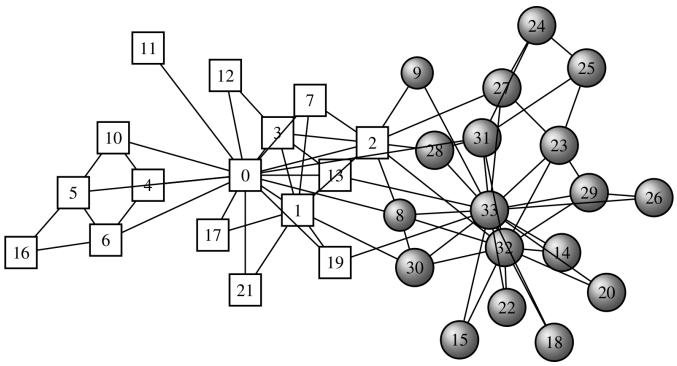

Figure 1: Zachary graph – The instructor and the administrator are represented by nodes 0 and 33. White squares represent individuals who ended up aligning with the club's instructor after the split, shaded circles those who aligned with the administrator.

5 Results

Newman has released his benchmark graphs in [28]. We have chosen the following graphs from his database to examine:

Zachary's karate club: social network of friendships between 34 members of a karate club at a US university in the 1970s.

Les Miserables: co appearance network of characters in the novel Les Miserables.

Books about US politics: A network of books about US politics published around the time of the 2004 presidential election and sold by the on-line bookseller Amazon.com.

Several researchers worked with these graphs, e.g. [17, 18, 19, 20, 21] from the perspective of community structure.

5.1 Zachary's karate club network

The first example is taken from one of the classic studies in social network analysis. In the late 1970s Wayne Zachary observed social interactions between the members of a karate club at an American university [27]. The club split in two as a result of an internal dispute, see Figure 1.

The critical node of the graph is the node 2, which (in real life) stayed with the node 0 (he was the instructor). It is critical, since many clustering methods fail to find the correct community it belongs to. Maximizing extended optimality performs well: in every case (from $c = 2$ to $c = 6$) node 2 was assigned correctly.

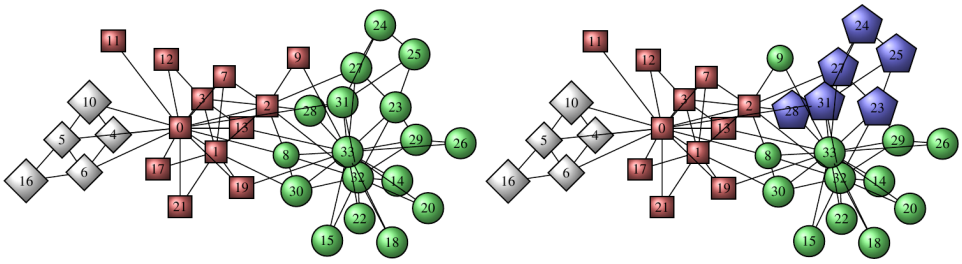

Figure 2: Zachary graph – The communities determined by the optimal extended modularity. Different shapes denote different community. (a) $c = 3$ (b) $c = 4$

Table 2: Zachary graph – membership vectors

Node	**Membership vector**
0 (inst)	(1.000000,0.000000,0.000000,0.000000,0.000000)
2	(0.985774,0.000000,0.000000,0.000000,0.014226)
9	(0.000000,0.014226,0.000000,0.000000,0.985774)
23	(0.000000,0.117156,0.882844,0.000000,0.000000)
32	(0.000000,1.000000,0.000000,0.000000,0.000000)
33 (adm)	(0.000000,1.000000,0.000000,0.000000,0.000000)

We tried a number of communities from $c = 2$ to $c = 6$. In these cases the optimal extended modularity determines a strict partition, except in the case $c = 5$. The nodes 2, 9 and 23 were assigned to more than one communities. All these nodes lie on the brink of two communities, see Table 2.

The result of the case $c = 3$ coincided with the result of label propagation method [25]. The optimum of the extended modularity in case $c = 4$ gave the same result as the walktrap method [23]. The case $c = 4$ gave the highest modularity (0.4198), above $c = 4$ the value of modularity started decreasing. See the modularity values in Table 3 .

5.2 "Les Miserables" graph

In the "Les Miserables" graph (see Figure 3) nodes represent characters in Victor Hugo's novel "Les Miserables" as indicated by the labels and edges connect any pair of characters that appear in the same chapter of the book [14].

The Table 5 is the summarizing table of the modularity values. In case $c = 4, 5$ and 8 the optimal extended modularity determines strict partitions again, but in the case $c = 11, 12$ we got some overlaps. In Table 4 the reader can see the non-binary membership vectors when the number of communities is 12.

Table 3: Modularity values – Zachary graph

Method	**Modularity**	**No. of comm.**
Edge betweenness	0.4013	5
Leading eigenvector	0.3727	3
Newman greedy	0.3807	3
Label propagation	0.4020	3
Spinglass	0.4063	6
Walktrap	0.4198	4
CPM ($k = 3$)	0.2438	3
CPM ($k = 4$)	0.2557	3
N^{++}	0.1947	12
Sequential greedy	0.3599	2
Optimum of the	0.4086	6
extended modularity	0.4159	5
	0.4198	4
	0.4020	3
	0.3718	2

Table 4: "Les Miserables" graph - Membership vectors of some nodes ($c = 12$)

Node	**Membership vector**
Valjean	(0.00, 0.00, 0.00, 0.00, 0.00, 0.31, 0.69, 0.00, 0.00, 0.00, 0.00, 0.00)
Marguerite	(0.06, 0.00, 0.21, 0.00, 0.00, 0.73, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00)
Isabeau	(0.00, 0.00, 0.00, 0.00, 0.00, 0.31, 0.00, 0.00, 0.00, 0.00, 0.69, 0.00)
Gervais	(0.00, 0.00, 0.00, 0.73, 0.00, 0.27, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00)
Tholomyes	(0.00, 0.00, 0.00, 0.00, 0.00, 0.25, 0.00, 0.00, 0.00, 0.75, 0.00, 0.00)
Scaufflaire	(0.00, 0.00, 0.00, 0.00, 0.00, 0.25, 0.00, 0.00, 0.00, 0.75, 0.00, 0.00)
Woman1	(0.00, 0.00, 0.00, 0.78, 0.00, 0.22, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00)
Boulatruelle	(0.00, 0.32, 0.00, 0.00, 0.00, 0.00, 0.68, 0.00, 0.00, 0.00, 0.00, 0.00)
Woman2	(0.00, 0.56, 0.00, 0.00, 0.00, 0.17, 0.00, 0.00, 0.00, 0.00, 0.00, 0.27)
MotherPlutarch	(0.00, 0.32, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.68, 0.00)
Toussaint	(0.00, 0.56, 0.00, 0.00, 0.00, 0.17, 0.00, 0.00, 0.00, 0.00, 0.00, 0.27)

Table 5: Modularity values – "Les Miserables" graph, *indicates the non-strict (fuzzy) partitions

Method	**Modularity**	**No. of comm.**
Edge betweenness	0.5381	11
Leading eigenvector	0.5116	4
Newman greedy	0.5006	5
Label propagation	0.5222	5
Spinglass	0.3809	12
Walktrap	0.5214	8
CPM ($k = 6$)	0.4041	4
CPM ($k = 7$)	0.4359	5
N^{++}	0.2946	16
Sequential greedy	0.4705	7
	0.4805	5
	0.4658	4
Optimum of the	0.4489*	12
extended modularity	0.5402*	11
	0.5453	8
	0.5562	5
	0.5218	4

Table 6: Modularity values – Graph of political books [15]

Method	**Modularity**	**No. of comm**
Edge betweenness	0.5168	5
Leading eigenvector	0.4516	4
Greedy	0.5020	4
Label propagation	0.4946	3
Walktrap	0.5253	4
CPM ($k = 3$)	0.4695	4
N^{++}	0.1656	34
Sequential greedy	0.4243	5
	0.4578	4
	0.3883	3
The optimum of the	0.5217	5
extended modularity	0.5254	4
	0.5269	3

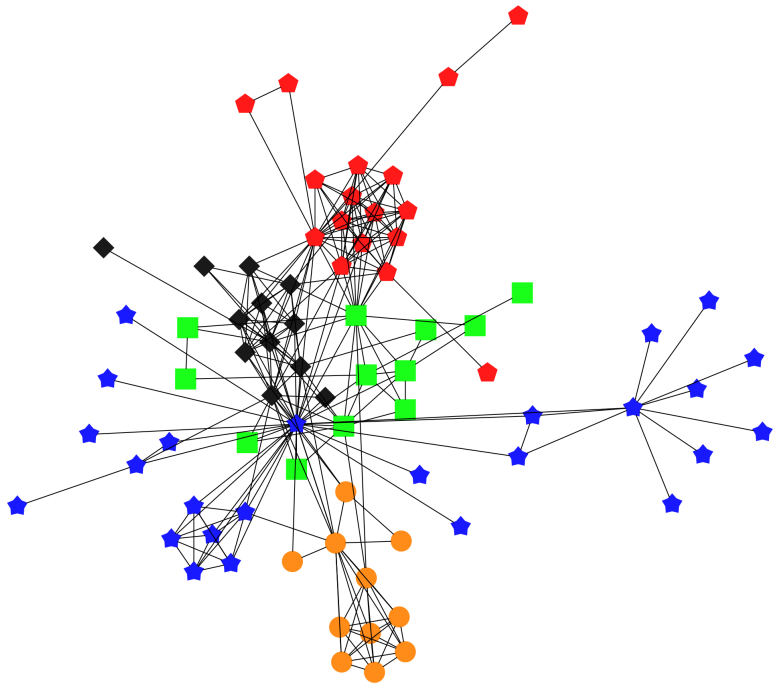

Figure 3: "Les Miserables" graph – Community structure with highest modularity ($c = 5$)

Table 7: Graph of political books – distribution of the same type books among communities

Community or cluster	*Edge betweenness*	*Eigen-*vector	*Walktrap*	*Label propagation*	*Newman greedy*	*CPM*
	liberal/neutral/conservative books					
0	3/2/2	0/1/2	5/4/3	4/4/2	5/4/3	1/2/0
1	0/3/42	0/2/35	0/2/39	0/7/46	0/6/43	0/4/44
2	39/2/1	43/7/3	38/2/1	39/2/0	38/2/1	43/4/6
3	0/4/4	0/3/9	0/5/6	–	0/1/2	0/4/4
4	1/2/0	–	–	–	–	–

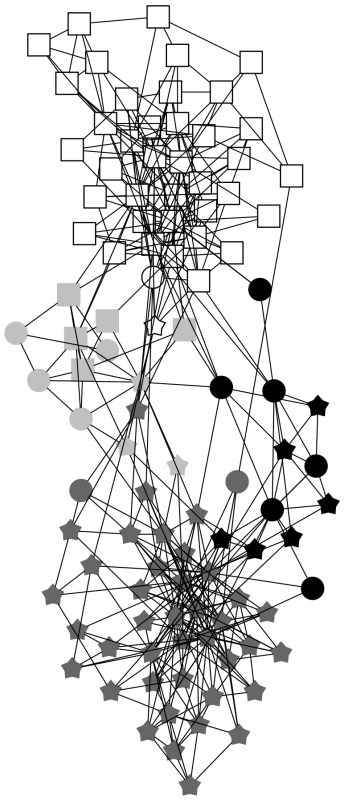

Figure 4: Political books graph – The communities in case $c = 4$, the modularity is 0.5254. The intensity shows the community, the shape indicates the original type of the book (square - liberal, circle - neutral, star - conservative).

5.3 Political books graph

The nodes in political books graph represent books about US politics sold by the online bookseller Amazon.com. Edges represent frequent co-purchasing of books by the same buyers, as indicated by the "customers who bought this book also bought these other books" feature on Amazon.

Nodes have been given values ℓ, n, or c to indicate whether they are *liberal*, *neutral*, or *conservative*. These alignments were assigned separately by Mark Newman based on a reading of the descriptions and reviews of the books posted on Amazon.

The optimum value of extended modularity in case $c = 3, 4$ and 5 gave strict partitions. In case $c = 4$ the optimal extended modularity gave almost the same division as the walktrap method, just one node was classified to another group, see Table 6.

The Table 7 shows the distribution of the book types among communities. It clearly shows, that every methods found two big and few (1, 2 or 3) small communities. One of the big communities contains mainly liberal books, the other contains conservative. The examination of the community/cluster 1 shows it mainly contains conservative book and few neutral, but none of the methods classified any liberal book to this class. As a contrast the community/cluster 2 mainly contains liberal books and some neutral, but almost every methods (except the label propagation) classified some conservative book in this community.

The rest of the clusters/communities are small and there are few which contains both liberal and conservative. The majority contains liberal and neutral or conservative and neutral books as we expect from a clustering method.

6 Conclusion

A new community detection method was proposed based on the definition of fuzzy partition and extended modularity. It was showed that the optimum of the extended modularity can determine overlapping and can give similar or sometimes better results, than earlier heuristics. However, it takes considerable more time to determine the optimum of the extended modularity, because it needs a solution of a quadratic optimization problem.

In solving QP problems it is important to find a good initial point. Thus we tried out the outputs of other methods proposed in [12, 22, 18, 19, 25] (see in Table 3, 5) as an initial point. Then the running time was reduced, but it is still too long. So the proposed method can be used only for small social networks.

The other proposed SG method provides good results measured in modularity, and its running time is acceptable as well (see Table 3, 5 and 6). In some case it gives higher modularity than other, more sophisticated methods (e.g. leading eigenvector method in Table 6). However it is strictly a clustering method, it cannot provide overlap.

There are still several ways for future work. We can define other similarity measures, or extend the modularity in some other reasonable way.

Other idea, to redefine the Equation 2, which is a constraint about the sum of the membership values of node i ($\forall 1 \leq i \leq n$). Instead of restrict this sum to be equal to 1, the sum can depend on the centrality of node i, so we can define it in the following way

$$\sum_{j=1}^{c} u_{ij} = g(k_i), \tag{8}$$

where $g(.)$ is an increasing function of the degree. This change allows for central nodes to have higher membership values. It is a rational change, because nodes with many connections usually belongs to more than one community.

We have already tested some cases on the Zachary graph. We increased the upper bound constraints of the sum of the membership values of the 3 main node: the 0, 32 and the 33. We can obtain from our observation that the increase of the upper bounds gives overlaps and maybe gives interesting results, thus it is a possible way for future work.

Acknowledgments. The second author was partially supported by the grants **TÁMOP-4.2.1/B-09/1/KONV-2010-0005, and OTKA K76099.**

References

[1] B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi, T. Vicsek, CFinder: Locating cliques and overlapping modules in biological networks. *Bioinformatics* **22**, 1021–1023 (2006).

[2] R. Albert and A. L. Barabási, Statistical mechanics of complex networks. *Reviews of Modern Physics*, **74** 2002.

[3] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Publishing, New York, NY, USA, 1981, ISBN 978-0306406713.

[4] J. C. Bezdek and S. K. Pal, *Fuzzy models for pattern recognition: methods that search for structures in data*, IEEE Press, New York, NY, USA, 1992

[5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks. arXiv:0803.0476v2 [physics.soc-ph]

[6] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski and D. Wagner, On Finding Graph Clusterings with Maximum Modularity. *In Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science* (WG'07) (2007)

[7] A. Clauset, M.E.J. Newman and C. Moore, Finding community structure in very large networks. *Physical Review E* **70**(066111) (2004)

[8] A. Csernenszky, Gy. Kovács, M. Krész, A. Pluhár, T. Tóth, *The use of infection models in accounting and crediting, Methods for Analyzing Social and*

Economical Processes, International Conference, University of Szeged, November 19 – 21, 2009.

[9] A. Csernenszky, Gy. Kovács, M. Krész, A. Pluhár: *Parameter Optimization of Infection Models*, $(CS)^2$ - Conference of PhD Students in Computer Science, Szeged, Hungary, June 29 – July 2, 2010

[10] L. Csizmadia: *Közösségfelismerés ismeretségi gráfokban [Community detection in social graphs]*, masterthesis, University of Szeged, 2003

[11] L. Csizmadia, A. Bóta and A. Pluhár, Community detection and its use in Real Graphs. *Proceedings of the 13th International Multiconference INFORMATION SOCIETY - IS* 2010, 393-396.

[12] I. Derényi, G. Palla, T. Vicsek: *Clique Percolation in Random Networks*, Physical Review Letters **94**, 160202:1–4. (2005),

[13] S. Fortunato: *Community detection in graphs*, Physics Reports **486**, 75–174 (2010),

[14] D. E. Knuth: The Stanford GraphBase: *A Platform for Combinatorial Computing*, Addison-Wesley, Reading, MA (1993)

[15] V. Krebs: unpublished, `http://www.orgnet.com/`

[16] T. Nepusz, A. Petróczi, L. Négyessy, F. Bazsó: *Fuzzy communities and the concept of bridgeness in complex networks*, Physical Review E **77** 016107 (2008),

[17] M.E.J. Newman: *Modularity and community structure in networks*, Proc. Natl. Acad. Sci. USA **103**, 8577–8582 (2006),

[18] M. Girvan, M. E. J. Newman: *Community structure in social and biological networks*, Proc. Nat. Acad. Sci. USA **99**, 7821–7826 (2002),

[19] M.E.J. Newman: *Finding community structure in networks using the eigenvectors of matrices*, Physical Review E **74**, 036104 (2006),

[20] M. E. J. Newman: *Fast algorithm for detecting community structure in networks*, Physical Review E **69**, 066133 (2004),

[21] V. Nicosia,G. Mangioni, V. Carchiolo and M. Malgeri: *Extending the definition of modularity to directed graphs with overlapping communities* J. Stat. Mech. (2009) P03024

[22] G. Palla, I. Derényi, I. Farkas and T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435**, 814 (2005).

[23] P. Pons, M. Latapy: *Computing communities in large networks using random walks*, Springer Berlin / Heidelberg, pp. 284–293,

[24] M. A. Porter, J.-P. Onnela and P. J. Mucha: Communities in Networks, *Notices of the American Mathematical Society* **56**, 1082–1097 & 1164–1166 (2009)

[25] U.N. Raghavan, R. Albert, S. Kumara: *Near linear time algorithm to detect community structures in large-scale networks*, Physical Review E **76**, 036106 (2007),

[26] J. Reichardt, S. Bornholdt: *Statistical Mechanics of Community Detection*, Physical Review E **74** (2006),

[27] Wayne W. Zachary: *An information flow model for conflict and fission in small groups*, Journal of Anthropological Research **33**, pp. 452–473 (1977)

[28] Network data `http://www-personal.umich.edu/~mejn/netdata/`

[29] igraph package `http://igraph.sourceforge.net/`

[30] graphopt algorithm `http://www.schmuhl.org/graphopt/`