Scalable Method for Information Spread Control in Social Networks

Michał Wojtasiewicz[1,2], Mieczysław Kłopotek[1] and Krzysztof Ciesielski[1]

[1] Institute of Computer Science, Polish Academy of Sciences,
ul. Jana Kazimierza 5, 01-248 Warsaw, Poland
[2] Warsaw University of Technology, Faculty of Mathematics and Information Science,
ul. Koszykowa 75, 00-662 Warsaw, Poland

Abstract. In this paper scalable and parallelized method for cluster analysis based on random walks is presented. The aim of the algorithm introduced in this paper is to detect dense subgraphs (clusters) and sparse subgraphs (*bridges*) which are responsible for information spreading among found clusters. The algorithm is sensitive to vertices assignment uncertainty. It distinguishes groups of nodes which form sparse clusters. These groups are mostly located in places crucial for information spreading so one can control signal propagation between separated dense subgraphs by using algorithm provided in this work. Authors have also proposed new coefficient which measures quality of given clustering in a sense of an information spread control between clusters. Measure presented in this paper can be used for determining quality of whole partitioning or a single *bridge*.

1 Introduction

The most common way to grasp real world phenomena is to distinguish objects (elements, entities) and relations (interactions etc.) between them. From a global perspective the entities and their relations form networks. The distribution of relations is usually not uniform and hence some structures of elements and their relations may be distinguished. Mining such structures has the potential of discovering new knowledge about the network or its parts or even members. One of the basic steps of network mining is splitting of the network, or rather its representation as a graph, into clusters. Clusters are groups of nodes which are interconnected in a way distinguishing them from the surrounding network. These groups or subsets of nodes are usually characterised by a denser set of relations. Such subsets (clusters) can be interpreted in various ways. This creates a necessity for algorithms that can cope with diversity of possible meanings.

Commonness of networks in everyday life (e.g. the Internet, data sets of citings) implies using advanced methods to analyze them. The most common and natural coding method for networks are graphs. Graph structure and the way of information spread in networks are the most interesting fields of research in social network community detection. In this paper scalable method of cluster analysis based on random walks is presented. The method divides a graph into subsets, where some of them can be used for information spread control. The main aim of the algorithm presented in this paper is to detect dense subgraphs which can be interpreted as tight communities and to detect relatively sparse subgraphs interconnecting them called *bridges*, as they are deemed to bridge information spread between the tight communities. The method provides clustering sensitive to vertices assignment uncertainty. As a result of a introduced Parallelized Locally Aggregated Random Walks (ParaLARW) algorithm one receives division which distinguishes sparse groups of nodes responsible for signal transfer between clusters.

2 Related Work

So far many algorithms for detecting communities in networks have been developed. Among most popular and most frequent used techniques one has to distinguish four categories: *bisection methods*, *hierarchical methods*, *combinatorical methods* and *spectral methods* [1]. Because of the diversity of cluster analysis problems, each of the areas is used in different situations. Choice of a method of identifying clusters should be made so that the available knowledge about the data could be used the most effectively. In practical tasks one mostly deals with very large graphs, which frequently consist of hundreds of thousands nodes and millions of edges. In such situations there is a limited number of methods which provide a solution in a reasonable time. This is because of complexity problems and difficulty of finding dense sets in large networks. An initial analysis, e.g. estimation of expected number of dense sets, is hard to perform as well. These are the reasons why hierarchic methods are mostly preferred in such situations. The most efficient algorithms operate on smaller sets and then aggregate results with a determined stop condition [2][3][4]. In this paper authors introduced a parallelized version of hierarchic, scalable algorithm of cluster analysis which was firstly proposed in [5]. This algorithm returns a very special partition. Among standard clusters one can distinguish subgraphs which are sparse and cannot be clearly assigned to any dense clusters. These special subgraphs enable control of signal propagation between clusters. Such a situation is connected to a feature of the MCL algorithm and it was fully discussed in section 2.1.
Many of articles speaking about modeling or controling the information spread in networks focus on greedy selection of vertices that have the highest influence in graph [6]. The main problem with such an approach is that user starts with one the most influential vertex and then searches for the most influential node in given neighbourhood. It can be easily seen that this kind of thinking produces very local results. Additionaly it is very probable that first most influential ver-

tex is deep in cluster. Finding few most influential nodes in social networks in that way does not solve problem of signal propagation between clusters. The author of [7] noticed that vertices of high degree gather more information in their neighborhood, while vertices of lower degree quickly transfer information inside the graph. Similar behavior occurs in clusters obtained via MCL algorithm, as we point at in 2.1. It was noticed, that in dense subsets information is transfered relatively fast. It happens because such subsets have many internal edges and a few on the outside. That creates the problem of communication between the clusters, which should be solved by initiating signal transfer on the boundaries of the clusters. This is what ParaLARW does. An interesting approach for identyfing influential veritices was presented in [8]. Authors analyzed dynamic social networks and they developed algorithm which assigns *dynamic influential value*. This coefficient is based on a probability of spreading influence through time. It is calculated in a greedy way so there is again problem with local optimum. Because it takes into account information from future states of a network it is useless in static analysis. In work [9] authors introduced approach in which there can be more than one type of influence. Every node can have a *opinion* which is continous function of time. Despite that interesting approach authors assume that influence of node is given by its degree. This assumption is too strong. It is easy to imagine that vertex can have small degree but signal started in this vertex will propagate very fast. This will happen in situation when such a node is connected to several dense clusters.
In every work mentioned above vertices were considered separately. Algorithm introduced in this paper provides division in a graph where some groups of vertices can be used for signal diffusion control.

2.1 MCL Feature

During work on the scalable modification of Markov Clustering Algorithm (MCL) very interesting feature was revealed. The figure 1 shows behaviour of the algorithm in situation of three vertices.
In the figure 1 result of the standard MCL algorithm can be seen. Despite that there is no distinction between these three nodes, the method has found two clusters. It happened because probability mass ran out very fast from vertex number *3* to other vertices. It should be noticed that there are several edges between vertex number *3* and vertices *1* and *2*. This is why the MCL method decided to mark vertex *3* as a different, in a sense of probability mass distribution, subgraph. Role of a node *3* is to transfer random walker between vertices *1* and *2*. It is not hard to imagine that nodes from figure 1 can be groups of vertices. If one of the groups hidden underneath vertex *3* forms a sparse cluster and its neighbouring clusters are dense then vertex *3* is a bridge candidate according to a definition from section 3. As can be seen in section 5.1 bridges play important role in information spread through networks. If one wants to reach as many units as possible in shortest possible time then it is not recommended to start in node located deep in a dense cluster. In such a situation signal will need a lot of time to travel between different clusters. The best way is to identify

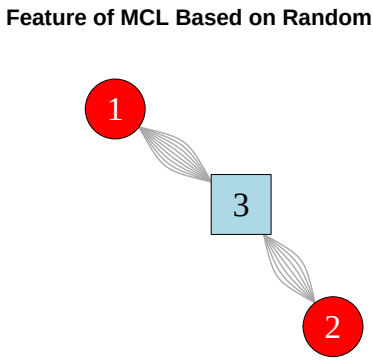


Fig. 1. MCL feature. Nodes 1 and 2 were pushed into one cluster and node 3 into a second one.

bridges (if they exist) and initate signal in one or some of them. On the other hand removing bridges adjacent to a considered cluster will make leaving that cluster more difficult. One can control signal propagation over a graph by simply opening and closing information flow through bridges.

3 ParaLARW Algorithm

Popular way of dealing with a complex problem is to divide it into smaller parts. The point of this process is to minimize the complexity without losing key data. One has to find optimal trade-off between global and local approach. The algorithm presented in this paper is an answer to a problem of scalability of MCL method [10]. That algorithm relies on simulating random walks on a network. The MCL procedure comes down to multiplication of stochastic matrices. There are several computational problems related. The most significant is multiplying very large matrices. Because of that one has to have huge amount of operation memory and computational power. At the beginning of the process stochastic matrix is sparse but it becomes dense quickly, after several steps. As the matrix gets more and more dense the operation memory starts to become insufficient. It regards even small graphs. The solution suggested in this paper is based on execution computations on specific subsets of graph. Dense subsets are separated using the MCL algorithm locally and then aggregation step is performed. This is a hierarchic method which results with multilevel clustering. Such a division has an important advantage. Among selected clusters are subgraphs which are not dense in a sense of a number of internal edges. Authors have named these sparse subgraphs *bridges candidates* which were defined in section 4.

3.1 Scheme

In this section authors recall a scheme of the LARW algorithm proposed in [5]. The scheme consists of three main steps which are discussed briefly below and can be seen in figure 2.

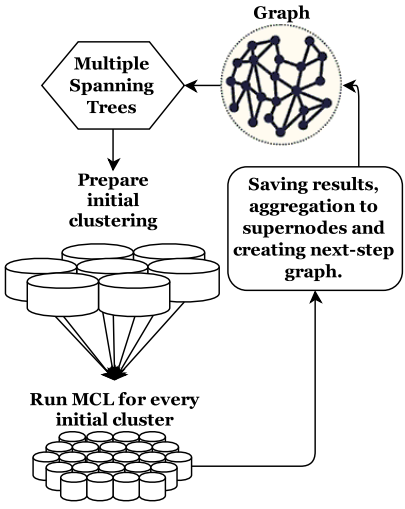


Fig. 2. Flowchart of LARW algorithm

1. Find spanning tree $T(G)$ of a given graph G. Now find vertex v which fulfills condition:

$$V(T(G))_{min} = \underset{u \in V(T(G))}{\operatorname{argmin}} \ (deg(u)) \tag{1}$$

$$v = \underset{w \in G}{\operatorname{argmax}}(deg(w) : w \in V(T(G))_{min}) \tag{2}$$

where $V(T(G))$ is set of all vertices in graph G and $deg(v)$ is a degree of vertex v. Next, cut out neighbourhood of rank r of found vertex v. Save the rest of a graph as G'. Repeat this step for all next G' until reaching situation when all nodes are assigned to some neighbourhood. This first clustering will be called *initial clustering*.

2. Apply the MCL method for every cluster in initial clustering. Save received results.

3. Aggregate every cluster from the second step to one *supernode*. Create a new graph from supernodes and assign transition probabilities between them as a sum of probabilities between vertices from considered clusters. Such a graph save as G''.

4. If number of edges in G'' (without loops) is equal to 0 then STOP otherwise $G = G''$ and go to step 1.

The ranks of neighbourhoods are chosen arbitrary once for all runs of the algorithm. In future authors will investigate procedure with adaptive choice of neighbourhoods orders.
The first step of the scheme above contains an important rule for choosing vertices. This rule should cause a situation where vertices chosen firstly are located near borders of clusters. Neighbourhood of that vertex probably consists of vertices from different clusters. Every neighbourhood requires utilization of one spanning tree in a graph so algorithm finds multiple spanning trees for multiple neighbourhoods of initial clustering.
The MCL algorithm should find out that certain initial cluster has to be divided according to borders of actual dense subsets.
Aggregation of clustering results with grouping vertices to a supernode is a typical technique of hierarchic algorithms. As one can see proposed algorithm returns multilevel clustering with more and more coarse partitioning. After one run of LARW [5] one can choose the most adequate clustering between several obtained. The main idea of the algorithm is to recognize where in the graph are located borders of true clusters and then divide initial clusters along them. Local approach satisfies requirement of scalability of algorithm for large datasets. Hierarchic way ensures that vertices near to a border which are from different clusters will be still separated. The LARW performs tens or hundreds times faster than MCL [10] for large sets and that advantage becomes higher with larger graphs.
In this paper the parallel version of LARW was introduced. Because of structure of the algorithm the second step (*Run MCL for every initial cluster*) from figure 2 can be easily done in parallel manner. Every run of MCL on any initial cluster is independent from the others so can be performed by different thread. Firstly introduced in [5] LARW algorithm is a single thread version of ParaLARW. Table 1 contains running times in seconds for every part of LARW and ParaLARW. Times were measured only for the most demanding first stage of algorithms. It means before any aggregation. As can be seen by parallelization of LARW one can achieve even 30% profit of a running time. Of course such method should be applied only to large graphs where time needed for proper initialization of parallel computations is significantly smaller than time needed for multiplying stochastic matrices.

4 Bridges Quality

In this section the new clustering quality coefficient is presented. Until now many different coefficients were developed and considered [11]. A significant number of them is concentrated on measuring how dense clusters are. One of them is the most frequently used. It is so called *modularity* [12]. Value of modularity measures how far from a random graph is certain cluster. In this paper authors

Graphs	Initial clustering	1 thread MCL	3 threads MCL	Aggregation
Snapshot	507.75	1006.25	694.6	359.82
Gnutella	270.46	30.32	22.64	129.95
HepPh	71.52	23.76	16.81	31.19
HepTh	51.81	20.87	14.62	22.37
Coautorship	36.21	15.92	13.46	3.03
Polblog	0.12	8.97	9.85	0.2
Football	0	0.28	0.58	0.01
Lesmis	0	0.2	0.52	0
Dolphins	0	0.2	0.55	0.2
Zachary	0.02	0.21	0.53	0.02

Table 1. Table of running times in seconds for every part of algorithm.

proposed a new coefficient which was developed to measure quality of found bridges. It is clear that some bridges will be more important in a sense of information spread in network. Proper measurement of bridges quality will allow to choose clustering which is the best for information diffusion control.
Well defined coefficient should take into account three important factors: a probability that signal will travel between two clusters through given bridge ($\mathbb{P}_{bridge}$), density of neighbouring clusters (ds) and fraction of bridge vertices which directly participate in signal transfer (F_{bridge}). Together with introduction of new coefficient, the definition of bridge is needed. Intuitively bridges are subgraphs which have fewer internal edges than external ones. Additionaly they have at least two neighbouring clusters and at least two of those clusters are dense. Of course bridges should connect dense clusters stronger than edges that connect them directly. Formally, they are specified by Defintion 1. as follows:

Definition 1.
Among clusters one can distinguish two categories:
1) dense cluster with $ds_C > 0.5$ where ds of cluster C *is defined as a number of edges within cluster divided by half of sum of degrees in the cluster.*
2) the remaining ones which may be called sparse ones.
Among the sparse clusters one can distinguish bridge candidates that is ones with two or mores neighbouring dense clusters. For the bridge candidates statistics $\mathbb{P}_{bridge}$, F_{bridge} and ds_{bridge} were defined as follows:

$$\mathbb{P}_{bridge} = \frac{E_{bridge}(C_1, C_2, \ldots, C_k)}{E_{inter}(C_1, C_2, \ldots, C_k) + E_{bridge}(C_1, C_2, \ldots, C_k)}, \tag{3}$$

$$F_{bridge} = \frac{V(bridge) - V(bridge)_{iso}}{V(bridge)}, \tag{4}$$

$$ds_{C_j} = \frac{E(C_j)}{\frac{1}{2}\sum_{v \in V(C_j)} deg(v)}, \tag{5}$$

$$ds_{bridge} = \frac{ds_{C_1} + ds_{C_2} + \ldots + ds_{C_k}}{k}, \tag{6}$$

where $E_{inter}(C_1, C_2, \ldots, C_k)$ *is a number of edges which directly connect dense clusters* $C_1, C_2, \ldots, C_k$ *(these clusters are all the dense neighbors of the bridge candidates),* $E_{bridge}(C_1, C_2, \ldots, C_k)$ *is a number of edges which connect bridge to its neighbouring dense clusters,* $V(bridge)_{iso}$ *is number of bridge candidate vertices which are not directly connected to dense clusters,* $V(bridge)$ *is a number of all vertices which form given bridge candidate and deg is a degree of a given vertex. Bridge candidate for which* $\mathbb{P}_{bridge} > 0.5$ *will be called bridge.*

All bridges considered in further part of this paper follow definition 1.
The measure of bridge quality, $InfoSpred_{bridge}$, will be introduced.

$$InfoSpread_{bridge} = \frac{3 \cdot \mathbb{P}_{bridge} \cdot ds_{bridge} \cdot F_{bridge}}{\mathbb{P}_{bridge} + ds_{bridge} + F_{bridge}}. \tag{7}$$

Intuitively, a dense cluster is one that would capture a random walker within it for a large number of steps. They may be viewed as tight communities. A bridge would participate in the transfer of a random walker between two dense clusters on the rare occasions when he leaves a dense cluster. $\mathbb{P}_{bridge}$ shows how often random walker will travel through bridge if such a transfer occurs. F_{bridge} tells about importance of internal nodes of bridge candidate for such a transfer. ds_{bridge} expresses how difficult it is for random walker to escape the dense clusters that bridge candidate connects. These three quantities characterize different aspects of being a connection between communities. All three measures range between 0 and 1. As a consequence, also $InfoSpread_{bridge}$ lies in the same range. To compare two different clusterings one can use sum of InfoSpread values over all bridges. Such a measure was shown in table 4. In future work authors will use values of InfoSpread not only to determine which bridge candidates are proper bridges but also to aggregate them. Such a procedure will result in a smaller number of bridges but characterised by more important role in information transfer. Rest of bridges candidates will be joined to dense clusters in order to maximize the value of modularity.

5 Results

5.1 Clustering Quality

In this section we compare the proposed ParaLARW algorithm with three methods: Louvain algorithm [2], Walktrap [3] and FastGreedy algorithm [4]. These three methods maximize modularity coefficient [12]. Louvain method is a hierarchic, greedy algorithm. It aggregates neighbouring vertices in order to maximize modularity. Walktrap works similarly to Louvain but distances between vertices are calculated using short random walks. Walktrap uses gain of modularity as a stop condition of aggregation step. The FastGreedy method is very similar to

Louvain algorithm. The main difference is that Louvain is multi-level (hierarchic) clustering method. Because of that FastGreedy results with slightly different partitioning. However, authors noticed that running time needed to receive results with the FastGreedy method is significantly smaller and this is because of tricky use of special data structures and clever algorithm implementation.
Because ParaLARW results in multilevel clustering, so partition which corresponds to the highest value of modularity has been chosen. We used for evaluation datasets listed in table 2 and there one can see number of vertices (*V(G)*) and edges (*E(G)*) of chosen graphs. All of datasets can be found on [13] or [14]. In table 4 authors presented four values: modularity [12], assortativity [11], number of found bridges and values of sum of InfoSpread Coefficient introduced in section 4. Because of definition 4 if clustering has higher number of true bridges it has higher potential of information spread over graph. Sum of InfoSpread is useful in case when there is similar number of true bridges in both compared partitionings. Average values of InfoSpread could be misleading because distributions of InfoSpread values which are skew. In future work authors will investigated more measures for describing bridges. These are important statistics which measure quality of resulted clusterings. Authors have investigated four values of rank r which determines neighbourhoods: 1, 2, 3, 4. Also four values of inflation were investigated. These values are 1, 1.25, 1.5, 2. Previous analysis showed that setting larger values of inflation results with very fine clustering. In table 4 one can see statistics of clustering which is characterized with the highest value of modularity.

Graphs	**#V(G)**	**#E(G)**
Slashdot	82144	500481
Gnutella	62586	147892
HepPh	34546	420877
HepTh	27769	352285
Coauthorship	16264	47594
Polblog	1490	16726
Football	115	615
Lesmis	77	254
Dolphins	62	159
Zachary	34	75

Table 2. Table of graphs descriptions.

As can be seen in table 4 for almost every graph, values of modularity and assortativity are lower in case of ParaLARW algorithm. Other three algorithms are aimed at modularity maximization so it is hard to compete in that field. However, after analysis of the last two columns (number of found bridges and sum of their InfoSpread values) of table 4 one will see that ParaLARW almost

always outperforms other algorithms. It is worth to mention that authors have chosen ParaLARW clustering with highest value of modularity. Probably there can be found a partitioning with lower modularity but higher value of the sum of InfoSpread Coefficient. Interesting thing is that Louvain and FastGreedy algorithm did not find almost any bridges at all. It is because they are greedy algorithms strictly aimed to modularity maximization. Second interesting observation is that if ParaLARW did not find bridge no other algorithm did. It is quite clear that for small graphs it is very difficult to distinguish any vertices which lie between subgraphs and transfer random walker between dense clusters. Only one algorithm is really devoted to detect bridges and it is Walktrap. This method is based on random walks as well but it is aimed to maximization of modularity rather than InfoSpread. However, Walktrap finds more bridges in case of HepTh network. It is related to the fact that Walktrap has tendency to result with clustering generated by neighbourhoods of hubs - vertices of a very high degree. After analysis of structure of networks HepTh and HepPh authors realized that these two citition networks have very skew degree distribution. It means that there are a few hubs and rest are vertices with relatively small degree. In such a situation the Walktrap distinguishes large subgraphs with many internal edges generated by hubs and at the same time cuts off vertices which are between neighbourhoods of these hubs. The way of finding bridges in case of ParaLARW and Walktrap are significantly different. It is because of that Walktrap uses short random walks only to calculate distances between pairs of nodes while ParaLARW simulates random walk for determining approximated transfer probability distribution. As was shown in section 2.1 the MCL algorithm is able to distinguish groups of vertices which are responsible for information transfer between dense clusters. In future authors will investigate different ways of determining initial clustering which has important role in applying MCL locally. Proposed method clearly finds proper bridges what is shown more precisely in section 5.2.

5.2 Signal Initialization

Possible way of analyzing bridges influence is to simulate how fast signal discovers a graph when it was started in a bridge against signal initialization inside a cluster. To do such simulation Markov Chain was involved again. For every bridge authors did the same procedure:

1. Identify bridge and cut out subgraph induced by vertices from considered bridge and adjacent clusters. Call that subgraph G_{sub}. Set of neighbouring clusters does not contain other bridges.
2. Simulate signal propagation by multiplying stochastic matrices (related to G_{sub}) 1,2,...,d times where d is diameter of G_{sub}. For every cluster and bridge in G_{sub}, in every step calculate how much of information has spread outside them in certain number of steps.

The simulation performed in step 2. was done by calculating fraction of positive transition probabilities from given cluster to the rest of analyzed subgraph G_{sub}.

Fractions where calculated for every step of a random walker. Now it is enough to compare fractions of positive probabilities derived from bridge and derived from its neighbouring clusters. To receive proper value for whole graph authors calculated two average fractions of positive probabilities, over all found bridges and over all neighbouring dense clusters, for every step of random walker. Every bridge has its G_{sub} with neighbouring dense clusters. Then for every step of random walker average differences of the fractions were calculated. Of course number of steps as well as the diameter can be different in different G_{sub}. Therefore calculated difference is a measure of an average coverage difference calculated for every step of random walker, for different signal initializations spots. The figure 3 shows results of ParaLARW bridges impact. Simulations were provided for datasets where at least one ParaLARW bridge was found.

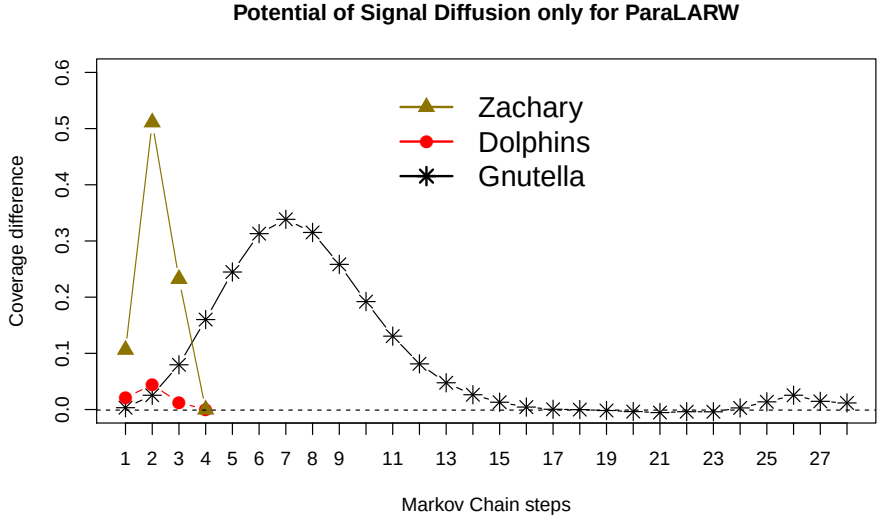


Fig. 3. Signal Initialization with ParaLARW

The figure 3 presents a potential of signal diffusion by using ParaLARW bridges. It proves that initialization of a signal in a bridge leads to higher coverage of a network than initialization in any other neighbouring dense cluster. All coverage differences are non-negative which means that signal started in a bridge explores graph faster. However, these graphs are only for academic research and are not interesting in a sense of practical usage. As can be seen in the figures 8, 4 random walker travels faster when the value of InfoSpread Coefficient is significantly higher. It means that proposed coefficient truly measures quality of a clustering with respect to its potential of information spreading. Clearly, with higher value of the InfoSpread for ParaLARW partitioning, faster exploration in several initial steps of networks Coauthorship and Slashdot is achieved. Such a behaviour

is preserved also for HepTh but with advantage for Walktrap(figure 5). In the case of HepPh (figure 6) the values of InfoSpread are very similar for Walktrap and ParaLARW but Walktrap bridges seem to be located more properly. It can be related to the fact that Walktrap results with larger clusters and it is very difficult to transfer random walker between dense clusters. Also Walktrap clustering has higher value of the modularity. Important conclusion appears. The ParaLARW does not work very well for citation networks. In near future it will be improved in this direction. The best performance of ParaLARW can be seen in the figure 7. The difference between proposed method and Walktrap is huge. It is worth mentioning that both sum of InfoSpread and modularity is higher in case of ParaLARW. There is no need to provide comparison with Louvain or FastGreedy algorithms because these methods hardly found any bridges and their values of InfoSpread are significantly smaller. Detailed analysis shows that sum of InfoSpread is related to dynamics of signal propagation in first steps while value of modularity determines maximal coverage differences. It is intuitive. If value of sum of InfoSpread is high so there is a lot of good bridges what means that signal started in in one of them will propagate quickly. In the same time value of modularity shows how well separated are dense clusters. In case of high value of modularity it is very hard for random walk to get out of dense cluster. That is why higher value of modularity implies higher maximal coverage difference. These two statistics should be considered jointly while analyzing bridges potential of information dispersion.

Both algorithms resulted with statistical significant difference in coverage but for almost every set confidence intervals are wider in case of Walktrap. Authors provided estimation quality with significance level equal 5%. Less acurate estimation for graphs HepPh and Coauthorship implies that there is hardly a difference between coverages achieved by ParaLARW and Walktrap, even if Walktrap results with higher mean. The rest of sets are characterized by good estimation with narrow confidence intervals.

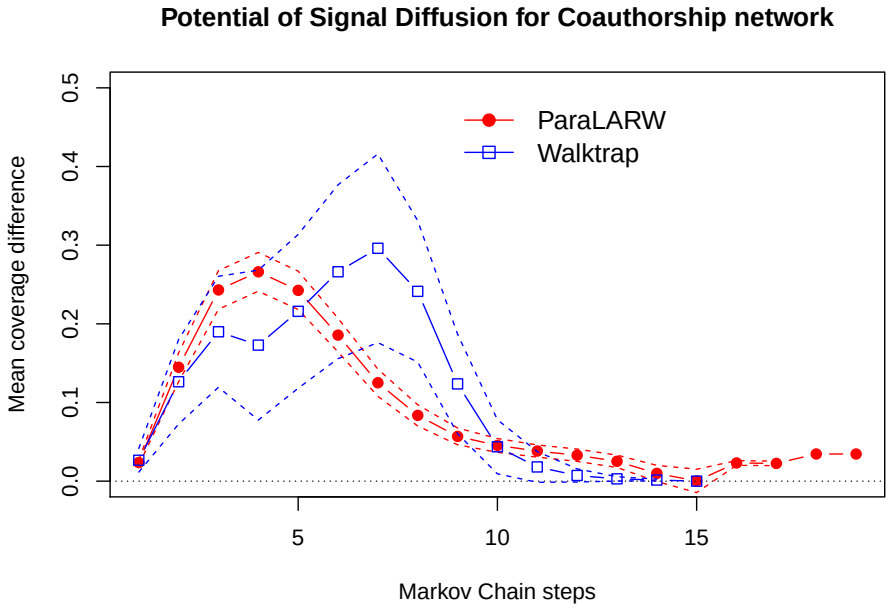


Fig. 4. Signal Initialization in Coauthorship network with ParaLARW and Walktrap

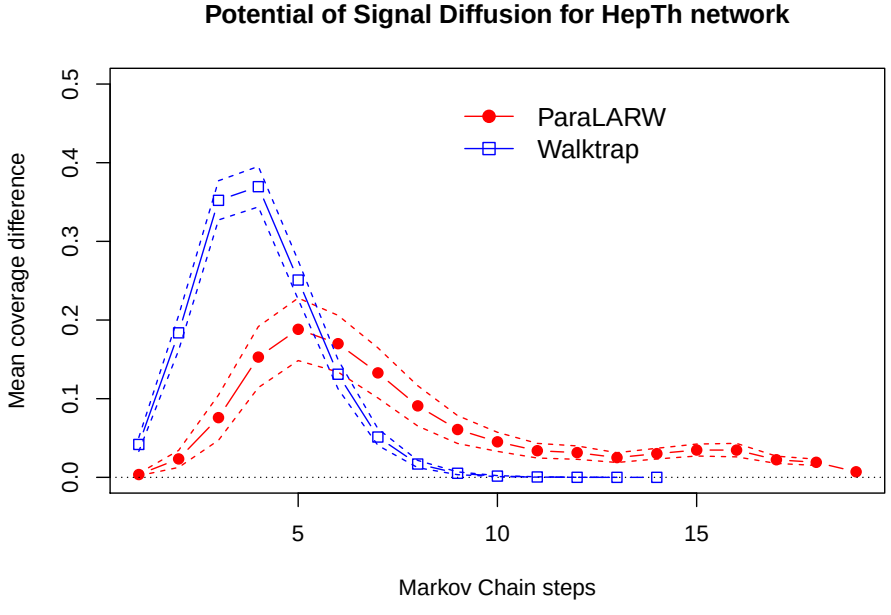


Fig. 5. Signal Initialization in HepTh network with ParaLARW and Walktrap

Graphs	Algorithm	Modularity	Assortativity	Found bridges	InfoSpread sum
Slashdot	Louvain	0.335	0.449	1	0.364
	ParaLARW	0.157	0.163	534	**102.31**
	Walktrap	0.186	0.265	123	66.03
	FastGreedy	0.319	0.469	1	0.327
Gnutella	Louvain	0.5	0.53	0	0
	ParaLARW	0.407	0.408	1501	**556.953**
	Walktrap	0.356	0.494	199	75.967
	FastGreedy	0.505	0.562	0	0
HepPh	Louvain	0.723	0.803	0	0
	ParaLARW	0.372	0.399	53	**27.1**
	Walktrap	0.681	0.775	41	26.036
	FastGreedy	0.526	0.857	0	0
HepTh	Louvain	0.657	0.723	0	0
	ParaLARW	0.328	0.38	71	39.55
	Walktrap	0.566	0.613	276	160.622
	FastGreedy	0.506	0.76	0	0
Coauthorship	Louvain	0.84	0.866	0	0
	ParaLARW	0.666	0.674	140	**83.2**
	Walktrap	0.742	0.78	20	10.128
	FastGreedy	0.78	0.873	0	0
Polblog	Louvain	0.427	0.843	0	0
	ParaLARW	0.412	0.828	0	0
	Walktrap	0.426	0.851	0	0
	FastGreedy	0.427	0.847	0	0
Football	Louvain	0.605	0.674	0	0
	ParaLARW	0.601	0.66	0	0
	Walktrap	0.603	0.671	0	0
	FastGreedy	0.55	0.671	0	0
Lesmis	Louvain	0.556	0.702	0	0
	ParaLARW	0.528	0.66	0	0
	Walktrap	0.521	0.714	0	0
	FastGreedy	0.5	0.652	0	0

Table 3. Table of clustering results.

Graphs	Algorithm	Modularity	Assortativity	Found bridges	InfoSpread sum
Dolphins	Louvain	0.519	0.679	0	0
	ParaLARW	0.455	0.538	1	**0.411**
	Walktrap	0.489	0.735	0	0
	FastGreedy	0.496	0.621	0	0
Zachary	Louvain	0.415	0.6	0	0
	ParaLARW	0.401	0.56	1	**0.595**
	Walktrap	0.346	0.545	0	0
	FastGreedy	0.387	0.621	0	0

Table 4. Table of clustering results.

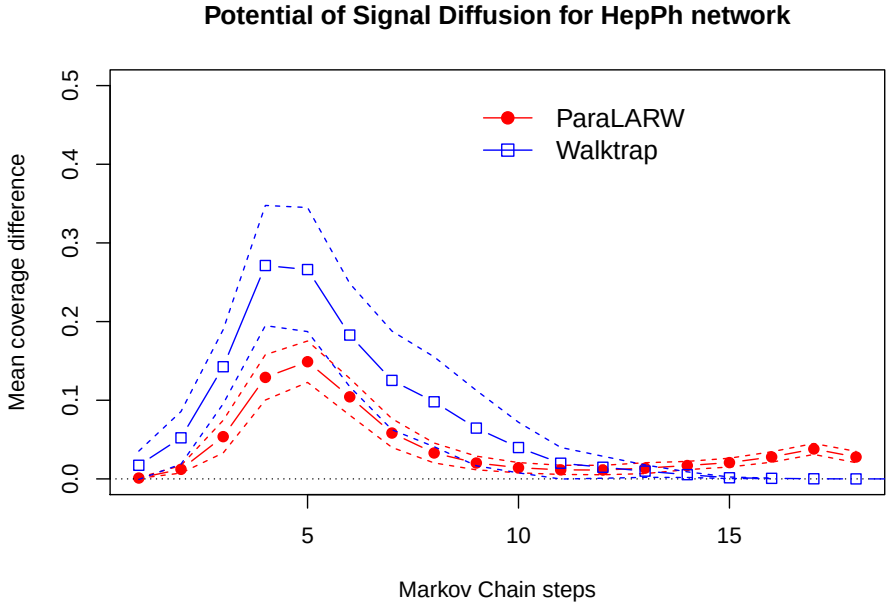


Fig. 6. Signal Initialization in HepPh network with ParaLARW and Walktrap

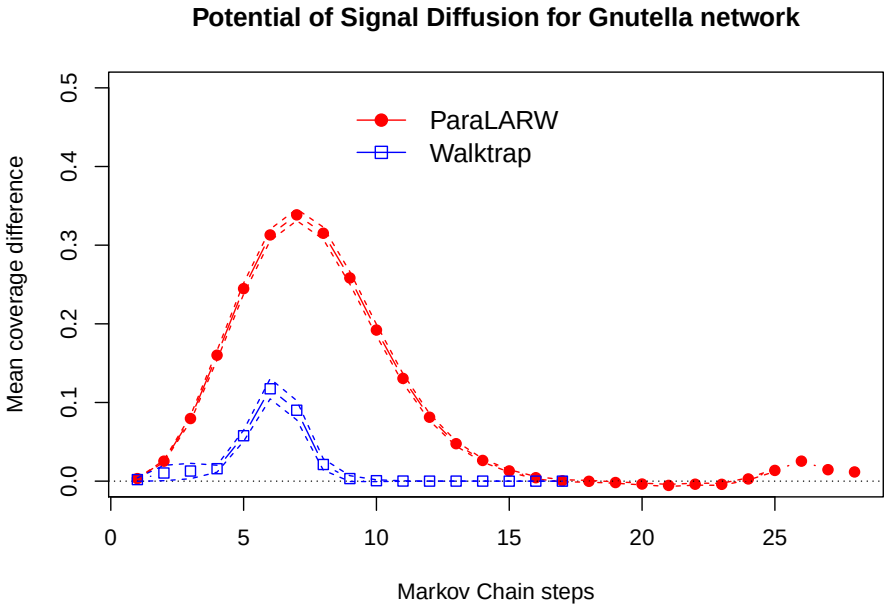


Fig. 7. Signal Initialization in Gnutella network with ParaLARW and Walktrap

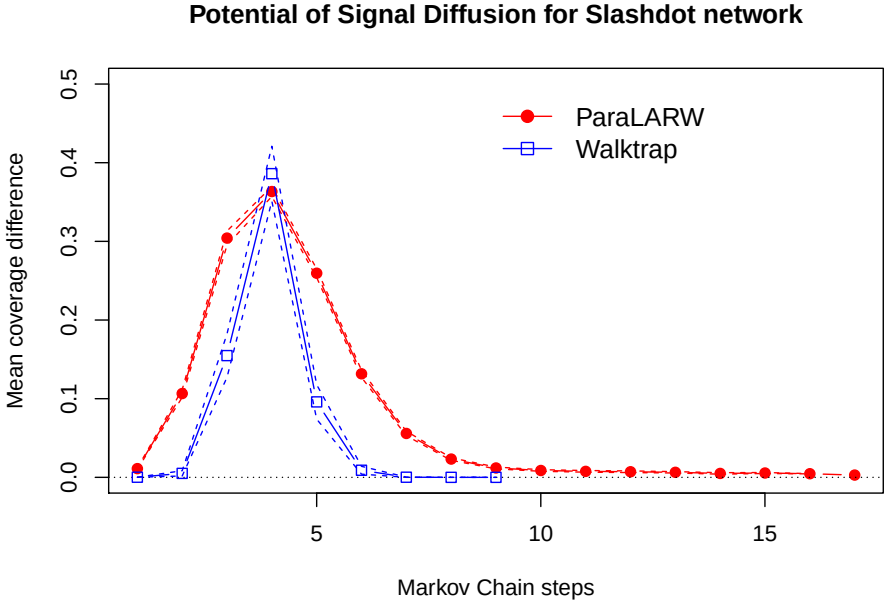


Fig. 8. Signal Initialization in Slashdot network with ParaLARW and Walktrap

5.3 Bridges characteristics

In this section authors described basic statistics of ParaLARW bridges for the most interesting graphs. Relationship between size of a bridge and InfoSpread Coefficent was analyzed as well.
Firstly it is important to check what percentage of bridge candidates is in fact true, useful bridges.

Graph	**Bridge candidates**	**True bridges**	**Percent**
Coauthorship	252	140	55.56%
HepTh	815	71	8.712%
HepPh	944	53	5.61%
Gnutella	1501	1501	100%
Slashdot	743	534	71.87%

Table 5. Table of fraction of true bridges

Graph	**Mean non-bridge candidate**	**Mean true bridge**	**Std.dev. non-bridge candidate**	**Std.dev. true bridge**
Coauthorship	0.247	0.594	0.108	0.15
HepTh	0.044	0.557	0.081	0.124
HepPh	0.039	0.517	0.069	0.136
Gnutella	0	0.371	0	0.081
Slashdot	0.092	0.156	0.028	0.062

Table 6. Table of means and standard deviations of InfoSpread Coefficient for bridge candidates and true bridges

As can be seen in table 5 only citation graphs have percentage of true bridges below 50%. The best result can be seen in row with Gnutella set what is very consistent with signal propagation from figure 7. Table 6 shows averages and standard deviations of InfoSpread. Average values were compared here because these are sparse clusters of different types - non-bridge candidates and true bridges. Non-bridge candidates cannot be used for information spread control because they do not transfer enough information. This is why sum of InfoSpread is meaningless in this situation. One can easily see that average values are significantly higher in the case of true bridges. More detailed comparison will be part of future work. Next interesting question is whether size of a bridge is correlated to value of InfoSpread Coefficient. Authors investigated occurence of such a dependency by constructing simple linear model and analyzing significance of

coefficient of explanatory variable which is InfoSpread. In the figure 9 one can see the relation between InfoSpread Coefficient and size of a bridge.

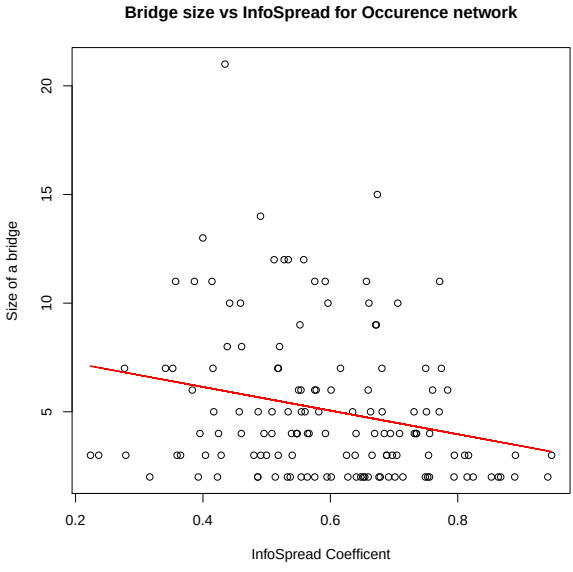


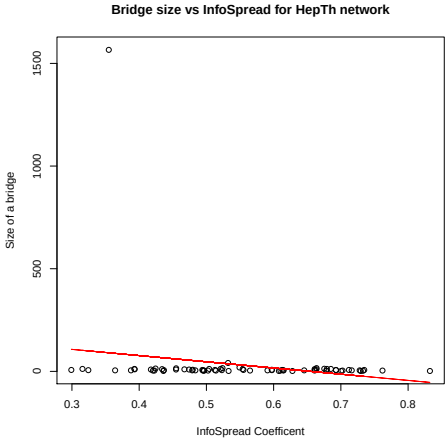
Fig. 9. Dependency between InfoSpread and size of a bridge for Couauthorship network

As can be seen in figure 9 InfoSpread Coefficent favours smaller bridges in case of Coauthorship set. Detailed statistics for all sets where placed in table 7.

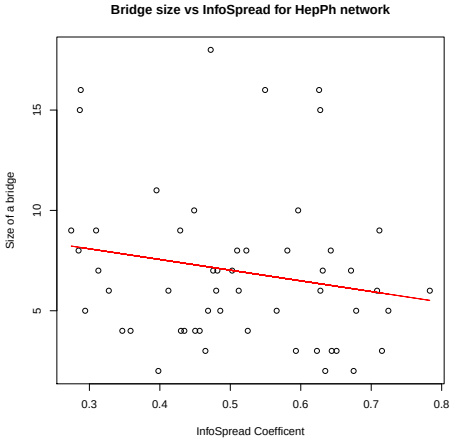
Graph	Coefficient	p-value
Coauthorship	-5.439	0.00462
HepTh	-302.3	0.0913
HepPh	-5.317	0.186
Gnutella	-66.73	0
Slashdot	-48.596	0

Table 7. Tabel of coefficients of linear models where InfoSpread was modeled by size of bridges.

Table 7 shows what can be seen in figure 9. Higher values of InfoSpread Coefficient correspond to smaller sizes of bridges. Taking 5% of significance level one can see that citations networks do not have this property. Figures 10a and 10b show relation in case of graphs HepTh and HepPh respectively.



(a) InfoSpread vs bridge size for HepTh



(b) InfoSpread vs bridge size for HepPh

Interesting observation is that structures of bridges in both cases are totally different. In case of HepTh all bridges are small except one. This single bridge on the other hand is very big. In second case sizes of bridges are very different and do not concentrate on small or high values of InfoSpread Coefficient.
Such a correlation between bridge size and InfoSpread can be induced by fact that InfoSpread consists partialy of information how many nodes of given bridge directly take part in signal transfer. In smaller bridges larger fraction of vertices will take part in signal propagation. That is why authors investigated correlation between size of a bridge and every component of InfoSpread Coefficient. For this purpose authors built one linear model for each set but this time every model has three explanatory variables. Each variable is one part of InfoSpread. Such a model will show which part is the most significant in sense of explaining size of a bridge.

Graph	**Coauthorship**		**HepTh**		**HepPh**	
	Coefficent	p-value	Coefficent	p-value	Coefficent	p-value
Probability	-4.175	0.03	-142	0.317	-2.974	0.33
Density	4.714	0.227	-706.9	0.026	-2.954	0.584
Fraction	-4.427	0.004	132.7	0.421	-3.631	0.391

Table 8. Tabel of detailed coefficients of linear models, part 1.

Tables 8 and 9 show very interesting results. Only in case of Coauthorship network part of InfoSpread which corresponds to fraction of nodes is significant with the 5% level of significance. The second very interesting observation is that in ev-

Graph	Gnutella		Slashdot	
	Coefficent	p-value	Coefficent	p-value
Probability	-371.273	0	-10.882	0
Density	-121.062	0	2.117	0.035
Fraction	-5.28	0.363	-0.655	0.513

Table 9. Tabel of detailed coefficients of linear models, part 2.

ery model coefficient related to probability that signal will flow through bridge is negative. It means that smaller bridges transfer more information between neighbouring dense clusters. This is exactly waht InfoSpread should measure. It should distinguish bridges which are small and very influential.
Conclusion of this section is that InfoSpread favours small bridges and for constant density and fraction, smaller bridges are characterized by higher probability of transfering information. In addition authors checked Pearson's correlation coefficient and any two pair of variables for any set was not correlated. For almost every pair correlation value was smaller then 0.3. Only for set Slashdot correlation between density and probability is equal 0.55, what is still not a high value. Of course because of high values of p-value, coefficients for HepTh and HepPh show that proposed values do not explain size of the bridges.

6 Conclusion

This section contains discussion of results received in sections 5.1 and 5.2. As can be seen in table 4 ParaLARW gives clustering with lower modularity, however it has a big advantage in detecting subgraphs responsible for information transfer between clusters. Authors have introduced new measure, InfoSpread Coefficient, which is useful for determining quality of found *bridges*. This new measure takes into account three most important properties of a bridge: $\mathbb{P}_{bridge}$, F_{bridge} and ds_{bridge}. InfoSpread can rank bridges for optimization purposes like a signal intialization cost.
In figures presented in section 5.2 one can clearly see that bridges have a serious impact on information dispersion over a network. Especially in case of large networks like Slashdot or Gnutella one can see that difference in coverage is significant. It can be seen that after several steps signal started in a dense cluster is trapped inside cluster while signal initialized in a bridge discoveres more and more nodes. After some steps coverages become similar but these first several steps are the most important. This is because of possible optimization of signal initialization cost and importance of time necessary for reaching certain number of nodes or communities.
It is clear that removing bridges from graph will significantly reduce potential of information transfer between clusters. This is because most of paths between dense clusters go through bridges.
ParaLARW (Parallelized version of LARW) is significantly faster (for large net-

works) than non-parallelized version. One can see in table 1 that running time gain can achieve even 30% for the most demanding stage of the ParaLARW clustering process.
Next modifications of ParaLARW will allow to achieve better results for citation networks which have very skew nodes degrees distribution.

6.1 Future Work

In this section future directions of research are presented.
As can be seen in table 1 time needed to perform the initial clustering can be really high in comparison to main part which is MCL run. One can solve this problem by providing adequate initial partitioning without building multiple spanning trees. Probably a proper sequence of neighbourhoods can be determined with just one spanning tree. The authors consider to investigate more ideas for providing initial clustering in the future.
The second direction of future work is proper aggregation of neighbouring bridges. As can be seen in table 4 in some cases ParaLARW finds a lot of bridges. Part of them could be aggregated to one bridge by maximization of InfoSpread Coefficient. Proper bridges should be maximally influential so appending two less important bridges to one more influential can have significant impact on information spread control.
Third important field of future research is different way of aggregation clusters to supernode. One can construct supernode from more than one cluster resulted from MCL. It could gain profit if one will aggregate more than one cluster what will maximize modularity. Another very important area of research is using random walks to detect bridges in any partitioning. If one has already derived clustering which satisifies some conditions then only need is to find proper bridges for information spread control. Such an approach will save time needed for reiteration of cluster analysis and will maintain important features of partitioning, like high value of modularity.

Acknowledgment

The paper is co-founded by the European Union from resources of the European Social Fund. Project PO KL “Information technologies: Research and their interdisciplinary applications”, Agreement UDA-POKL.04.01.01-00-051/10-00.

References

1. Fortunato, S.: Community detection in graphs. Complex Networks and Systems Lagrange Laboratory ISI Foundation (2010)
2. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. P10008 (2008)
3. Pons, P., Latapy, M.: Computing communities in large networks using random walks. arXiv:physics/0512106 (2005)

4. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E 70, 066111 (2004) (2004)
5. Wojtasiewicz, M., Ciesielski, K.: Identifying bridges for information spread control in social networks. Workshop on Social Influence September 10 2014 Barcelona Spain. SocInfo 2014 Workshops LNCS 8852 pp. 390–401 2015 (2014)
6. Kempe, D., Kleinbergy, J., Tardosz, E.: Influential nodes in a diffusion model for social networks. LNCS Volume 3580, pp 1127-1138 (2005)
7. Doerr, B., Fouz, M., Friedrich, T.: Why rumors spread fast in social networks. Communications of the ACM, Vol. 55, No. 6, pp. 70-75 (2012)
8. Aggarwal, C., Lin, S., Yu, P.S.: On influential node discovery in dynamic social networks. SDM, 636-647 (2014)
9. Ju, C., Cao, J., Zhang, W., Ji, M.: : Influential node control strategy for opinion evolution on social networks. Abstract and Applied Analysis, Article ID 689495 (2013)
10. van Dongen, S.M.: Graph clustering by flow simulation. PhD thesis, Universiteit Utrecht (2000)
11. da F. Costa, L., Rodrigues, F.A., Travieso, G., Boas, P.R.V.: Characterization of complex networks: A survey of measurements. Advances in Physics, Volume 56, pages 167 - 242, Issue 1 (2007)
12. Newman, M.E.J.: Modularity and community structure in networks. PNAS 103 (23) 8577-8582 (2006)
13. Newman, M.: http://www-personal.umich.edu/ mejn/netdata/ (2012)
14. University, S.: http://snap.stanford.edu/data/ (2012)