Detecting communities in large networks using the extended Walktrap algorithm

DO Duy Hieu
Institute of Mathematics
Vietnam Academy of Science and Technology
Ha Noi, Viet Nam
ddhieu@math.ac.vn

PHAN Thi Ha Duong
Institute of Mathematics
Vietnam Academy of Science and Technology
Ha Noi, Viet Nam
phanhaduong@math.ac.vn

***Abstract*—In this paper, we extended the Walktrap algorithm [16], which improves the algorithm's performance. First, we propose an extended lazy random walk (ELRW), and then we use this walk to define the distance between vertices on a graph similar to the Walktrap algorithm. When applying this algorithm, if we choose the coefficient of a reasonably extended lazy random walk, we will get better clustering results when using the Walktrap algorithm. Finally, we also program to demonstrate the efficiency of this algorithm.**

I. Introduction

In recent decades, network analysis has gained considerable attention thanks to its wide application in many different domains such as social study (friends network) [13], biology (protein network) [17], [19] or computer science (www network, social network) [14]. Mathematically, a network is represented by a graph that consists of vertices connecting by edges. Although graphs in most real-life cases are significant and sparse (they may contain up to a hundred thousand or even millions of vertices), there exist inside the community structures. A community can be understood as a group of similar vertices linked densely but loosely to the outer ones. Since this structure can be exploited thoroughly to gather information about the original network, it has become an intriguing topic for scientists in recent years.

Among numerous studies that have been made to solve the community detection problem, the spectral and dynamic approaches caught our special attention. It is not easy to mathematically define a community of a graph. In fact, several definitions were proposed in network studies, but they are weak one way or the other: too restrictive, cannot be computed efficiently, and so on. Many recent has exploited random walks to understand the structural properties of networks. Aditya and Jure [2] proposed a biased random walk procedure in their machine learning algorithm *node2vec*. Zhou and Lipowsky [20] also used a biased random walker, which usually moves towards the vertices which have maximum neighbors with the starting node in a hierarchical algorithm (called *Netwalk*). Markov Cluster Algorithm [20] iterates two matrix operations (one corresponding to random walks), bringing out clusters in the limit state.

Despite that, most recent approaches have reached a consensus and consider that a partition of vertices in a graph represents a good community structure if the Modularity is high. Newman proposed in [7] a greedy algorithm that starts with n communities corresponding to the vertices and merges communities to optimize the modularity function, which measures the quality of a partition. This algorithm runs in $O(n^2)$ and has recently been improved to a complexity $O(mH \log n)$ [21] where H is the depth of the dendrogram. Other exciting methods have been proposed; see, for instance, [3], [5], [6], [18].

A. Latapy and Pons work

In [16], the authors have designed the Walktrap algorithm that generates partitions from the given undirected and connected graph and evaluates the precision of these partitions by computing the Modularity. Walktrap algorithm uses agglomerative hierarchical clustering method [10], [11] together with random walks to detect intrinsic community structures. This clustering method iteratively groups the vertices into communities (with preset conditions) based on a measurement of the similarity between vertices.

More specifically, given a connected and undirected graph G, P denotes the transition matrix, and D denotes the diagonal matrix of vertices degrees. First and foremost, the following distance function has to be computed for all pairs of vertices

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P^t_{ik} - P^t_{jk})^2}{d(k)}} = \|D^{-1/2}P^t_{i\bullet} - D^{-1/2}P^t_{j\bullet}\|. \tag{I.1}$$

Then, based on Ward's method, which aims at minimizing the mean of squared variances function

$$\sigma_k = \frac{1}{n} \sum_{C \in \mathcal{P}_k} \sum_{i \in C} r^2_{iC}, \tag{I.2}$$

the algorithm merges the communities one by one. As this problem is NP-hard, the algorithm will instead merge two communities C_1 and C_2 into C_3 that minimize the increment

$$\Delta\sigma(C_1, C_2) = \frac{1}{n} \frac{|C_1||C_2|}{|C_1| + |C_2|} r^2_{C_1C_2}. \tag{I.3}$$

After the community merging step, the probability matrix and the $\Delta\sigma$ data are recalculated, and the process repeats until the whole network is put in a single community.

To estimate the quality of each partition, the authors proposed a quantity called $\eta_k = \frac{\Delta\sigma_k}{\Delta\sigma_{k-1}}$ besides the modularity function Q [10]. This function is calculated every time a new partition is formed to identify the best partition with the highest η value.

It is worth noticing that the algorithm can handle many practical big network situations in which the network could contain up to millions of vertices. This is due to the use of *random walk* in estimating the matrix P^t.

B. Our work

Our improvement for the Walktrap algorithm is to extend the algorithm by defining new random walks, namely the extended lazy random walk. And then, we induce a new distance, which is also closely related to the spectral properties of the transition matrix P. Next, we use the Walktrap algorithm with our new distance.

C. Preliminaries

Let $G = (V, E)$ denote an undirected and connected graph where V and E are the sets of vertices and edges, respectively. Denote $|V|$ by n, $|E|$ by m and the adjacency matrix of G by A. Here, $A_{ij} = 1$ if vertices i and j are connected (there is an edge between them) and $A_{ij} = 0$ otherwise. The degree $d(i) = \sum_j A_{ij}$ of vertex i is the number of its neighbors (including itself). For simplicity, we only consider unweighted graphs in this paper. It is, however, trivial to extend our results to weighted graphs ($A_{ij} \in \mathbb{R}^+$ instead of $A_{ij} \in \{0, 1\}$), which is another advantage of this approach.

We consider a random walk $X = X_0, X_1, \ldots, X_t, \ldots$ on this G. (see [9], [17] for a complete presentation of the topic). At each step t, a walker moves to a vertex chosen randomly and uniformly among its neighbors. The sequence of visited vertices is thus a Markov chain, whose states are the graph's vertices. At each step, the transition probability from vertex i to vertex j is $P_{ij} = \frac{A_{ij}}{d(i)}$. This defines the transition matrix P of the random walk process. Clearly, $P = D^{-1}A$ where D is the diagonal matrix of the degrees ($D_{ii} = d(i)$ and $D_{ij} = 0$ for $i \neq j$).

Remark 1.1: The t^{th} power of P contains information of the probability of moving from vertice i to vertice j after t steps. This value is exactly P^t_{ij}.
Another remark at the heart of our proposed models is as follows.

Remark 1.2: Let G be a connected, undirected graph with m communities. Then, the coordinates of the first $m - 1$ vectors show community structure, while the others, especially the close-to-zero and negative ones, are identified as "noises."

II. The extended Walktrap algorithm

One of the well-studied random walk models on a graph is the *lazy random walk*. This model considers the chances that the walker stays, which is possible in realistic situations. At each step, the walker will choose uniformly between staying at the same spot or visiting one of its neighbors. In this section, we try to extend this model by introducing a parameter μ that regulates the behavior of the *lazy walk*.

Let $\mu \in \mathbb{R}$ ($0 \leq \mu \leq 1$). We introduce a modified version of the original walk, which we call *extanded lazy random walk*. In a extanded lazy random walk, at time t:

- the walker moves with probability $\frac{1}{1+\mu}$,
- the walker stays at the current vertex with probability $\frac{\mu}{1+\mu}$.

In case $\mu = 1$, the *extanded lazy random walk* becomes the *lazy random walk*, and with $\mu = 0$, *extanded lazy random walk* becomes *simple random walk*. We can show that the above modification breaks the periodicity of the random walk, thus making the random walk applicable to the case of bipartite graphs. The transition probabilities are encoded in the following matrix:

$$\mathcal{P} = \frac{1}{1+\mu}(\mu I + P), \tag{II.1}$$

where I denotes the identity matrix. In an analogous manner to the original *Walktrap* algorithm, we give the formula for the distance function that measures the vertices' similarities

Definition 2.1: Let i and j be two vertices in the graph and

$$\mathcal{R}_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(\mathcal{P}^t_{ik} - \mathcal{P}^t_{jk})^2}{d(k)}} = \|D^{-1/2}\mathcal{P}^t_{i\bullet} - D^{-1/2}\mathcal{P}^t_{j\bullet}\|. \tag{II.2}$$

Now, we shall inspect the analytic aspect of this model. First of all, we need the following lemma

Lemma 2.2: ([16, Lemma 1]) The eigenvalues of the matrix P are real and satisfy:

$$1 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_n \geq -1. \tag{II.3}$$

Moreover, there exists an orthonormal family of vectors $(s_\alpha)_{1\leq\alpha\leq n}$ such that each vector $v_\alpha = D^{-1/2}s_\alpha$ and $u_\alpha = D^{1/2}s_\alpha$ are respectively a right and a left eigenvector associated to the eigenvalue λ_α:

$$\forall\alpha,\ Pv_\alpha = \lambda_\alpha v_\alpha \ \text{ and } P^T u_\alpha = \lambda_\alpha u_\alpha$$
$$\forall\alpha, \forall\beta,\ v_\alpha^T u_\beta = \delta_{\alpha\beta}$$

From Lemma 2.2 and (II.1), we have the following lemma.

Lemma 2.3: The eigenvalues of the matrix $\mathcal{P}$ are $\gamma_\alpha = \frac{\lambda_\alpha+\mu}{1+\mu}$, $1 \leq \alpha \leq n$, and the vectors $v_\alpha = D^{-1/2}s_\alpha$ and $u_\alpha = D^{1/2}s_\alpha$ (defined in Lemma 2.2) are also respectively a right and a left eigenvector associated to the eigenvalue γ_α of matrix $\mathcal{P}$.

Proof From (II.1), we have

$$\mathcal{P}v_\alpha = \frac{1}{1+\mu}(\mu I + P)v_\alpha = \frac{1}{1+\mu}(\mu v_\alpha + Pv_\alpha), \tag{II.4}$$

From Lemma 2.2, we have $Pv_\alpha = \lambda_\alpha v_\alpha$, it is follow that

$$\mathcal{P}v_\alpha = \frac{1}{1+\mu}(\mu v_\alpha + \lambda_\alpha v_\alpha) = \frac{\mu+\lambda_\alpha}{1+\mu}v_\alpha, \tag{II.5}$$

Similarly, we have

$$\mathcal{P}^T u_\alpha = \frac{1}{1+\mu}(\mu I + P^T)u_\alpha = \frac{1}{1+\mu}(\mu u_\alpha + P^T u_\alpha). \tag{II.6}$$

From Lemma 2.2, we have $P^T u_\alpha = \lambda_\alpha u_\alpha$, it is follow that

$$\mathcal{P}^T u_\alpha = \frac{1}{1+\mu}(\mu u_\alpha + \lambda_\alpha u_\alpha) = \frac{\mu+\lambda_\alpha}{1+\mu}u_\alpha. \tag{II.7}$$

It is followed that the eigenvalues of the matrix $\mathcal{P}$ are $\gamma_\alpha = \frac{\lambda_\alpha+\mu}{1+\mu}$, $1 \leq \alpha \leq n$, and the vectors $v_\alpha = D^{-1/2}s_\alpha$ and $u_\alpha = D^{1/2}s_\alpha$ (defined in Lemma 2.2) are also respectively a right and a left eigenvector associated to the eigenvalue γ_α. □

Theorem 2.4: The distance $\mathcal{R}$ is related to the spectral properties of the matrix P by:

$$\mathcal{R}^2_{ij} = \sum_{\alpha=2}^{n}\left(\frac{\lambda_\alpha+\mu}{1+\mu}\right)^{2t}(v_\alpha(i) - v_\alpha(j))^2. \tag{II.8}$$

where $(\lambda_\alpha)_{1\leq\alpha\leq n}$ and $(v_\alpha)_{1\leq\alpha\leq n}$ are respectively the eigenvalues and right eigenvectors of the matrix P.

Proof Lemma 2.2 makes it possible to write a spectral decomposition of the matrix $\mathcal{P}$

$$\mathcal{P} = \sum_{i=1}^{n}\gamma_\alpha v_\alpha u_\alpha^T, \ \text{ and } \mathcal{P}^t = \sum_{i=1}^{n}\gamma_\alpha^t v_\alpha u_\alpha^T. \tag{II.9}$$

It follows that

$$\mathcal{P}^t_{i\bullet} = \sum_{i=1}^{n}\gamma_\alpha^t v_\alpha(i)u_\alpha = D^{1/2}\sum_{i=1}^{n}\gamma_\alpha^t v_\alpha(i)s_\alpha. \tag{II.10}$$

We put this formula into the second definition of $\mathcal{R}_{ij}$ given in Equation (2.1). Then we use the Pythagorean theorem with the orthonormal family of vectors $(s_\alpha)_{1\leq\alpha\leq n}$, and we remember that the vector v_1 is constant to remove the case $\alpha = 1$ in the sum. Finally, we have:

$$\mathcal{R}^2_{ij} = \sum_{\alpha=2}^{n}\gamma_\alpha^{2t}(v_\alpha(i) - v_\alpha(j))^2. \tag{II.11}$$

From Lemma 2.3, we have $\gamma_\alpha = \frac{\lambda_\alpha+\mu}{1+\mu}$, $1 \leq \alpha \leq n$, it follows that

$$\mathcal{R}^2_{ij} = \sum_{\alpha=2}^{n}\left(\frac{\lambda_\alpha+\mu}{1+\mu}\right)^{2t}(v_\alpha(i) - v_\alpha(j))^2. \tag{II.12}$$

□

III. Observations

We present some crucial observations that must be considered in using random walks to detect community structure. In addition, we give theoretical comparisons between the *simple random walk* and the *extanded lazy random walk*.

Remark 3.1: The transition matrix P has real eigenvalues satisfying:

$$1 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_n \geq -1.$$

Remark 3.2: In [4], let G be a network with apparent community structure. Then, P also has $m-1$ eigenvalues that are approximately 1, where m is the number of well-defined communities.

Remark 3.3: In [4], the eigenvectors associated to these first $m-1$ nontrivial eigenvalues also have a community characteristic, i.e., the elements that correspond to vertices within the same community are roughly the same.

We restate the result of M. Latapy and P. Pons in [16].

Theorem 3.4: ([16, Theorem 1]) The distance r is related to the spectral properties of the matrix P by:

$$r_{ij}^2 = \sum_{\alpha=2}^{n} \lambda^{2t} (v_\alpha(i) - v_\alpha(j))^2. \qquad \text{(III.1)}$$

where $(\lambda_\alpha)_{1 \leq \alpha \leq n}$ and $(v_\alpha)_{1 \leq \alpha \leq n}$ are respectively the eigenvalues and right eigenvectors of the matrix P.

The following table summarizes the results we have obtained in Theorems 2.4 and 3.4 (Note: SW is simple walk).

	The distances	$f_i(\lambda_\alpha)$
SW	$\sum_{\alpha=2}^{n} f_1(\lambda_\alpha)(v_\alpha(i) - v_\alpha(j))^2$	$f_1(\lambda_\alpha) = (\lambda_\alpha)^{2t}$
ELRW	$\sum_{\alpha=2}^{n} f_2(\lambda_\alpha)(v_\alpha(i) - v_\alpha(j))^2$	$f_2(\lambda_\alpha) = \left(\frac{\lambda_\alpha+\mu}{1+\mu}\right)^{2t}$

It can be seen that all of the above distances have $(v_\alpha(i) - v_\alpha(j))^2$ as their common factor, only differing in terms of $f_1(\lambda_\alpha)$ and $f_2(\lambda_\alpha)$. This fact, together with Remark 3.1, Remark 3.2 and Remark 3.3 show that one needs to weaken (ideally, eliminate) the ”noisy” eigenvalues, i.e., the close-to-zero or negative ones to yield a good graph partition.

In most cases, since $trace(P) = 0$ and P has some close-to-one eigenvalues, there will be eigenvalues that are close to -1. One example of this fact is the case of the bipartite graph, where the spectrum P is symmetric at about 0. We note that:

Remark 3.5: With the simple random walk, the eigenvalues with absolute values close to one change much ”slower” than those closer to zero when we take power t of P. Therefore, taking P^t $(3 \leq t \leq 8)$ will emphasize the more significant eigenvalues by wiping out the small ones.

Remark 3.6: With the extended lazy random walk, μ can be chosen appropriately so that the significant eigenvalues stay almost unchanged, while other ”noises” are depleted (going to zero) quickly through f_2. For example, $\mu \in (0.4, 0.5)$ for small graph, and $\mu \in (0.1, 0.2)$ for large graph.

Remark 3.7: For bipartite graphs, a simple random walk is due to the fact that the corresponding Markov process is periodic.

Example 3.8: Let the graph G as shown in the Figure 1, with $t = 1$, $\mu = 0.5$, we have graph of a function f_1 and f_2 as shown in the Figure 2. we see that the small and negative eigenvalues almost disappear or weaken in the first and second diagrams.

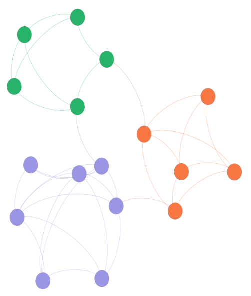

Fig. 1. An undirected, connected graph G

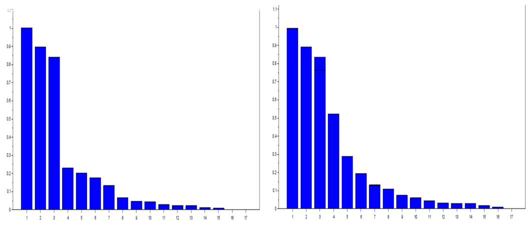

Fig. 2. The graph showing $f_1(\lambda)$ and $f_2(\lambda)$

IV. Examples

To evaluate the algorithm's efficiency, we use Newman's Modularity Q [15]. The expected number of edges falling between two vertices i and j in the configuration model is equal to $d_i d_j/2m$, where d_i is the degree of vertex i and m is the total number of edges in the observed network. The actual number of edges observed to fall between the same two vertices is equal to the element A_{ij} of the adjacency matrix A, so that the actual-minus-expected edge count for the vertex pair is $A_{ij} - d_i d_j/2m$. Giving integer labels to the groups in the proposed network division and denoting by g_i the label of the group to which vertex i belongs, the modularity Q is then equal to

$$Q_u = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta_{g_i g_j} \qquad \text{(IV.1)}$$

where δ_{ij} is the Kronecker delta. We note that, in the clustering results, the one that gives us the more significant Modularity Q, the better the clustering result.

Example 4.1: We consider an undirected network with 198 vertices. Then apply the Walktrap algorithm and our extended Walktrap algorithm with $\mu = 0.1$. Then we get the results as below.
We apply both algorithms with $t = 2$ and 4. For our algorithm, we take $\mu = 1$, and we obtain the corresponding Modularity values in table I. Figure 3, illustrates the clustering results of both algorithms with $t = 4$.

TABLE I
Modularity value comparison table

Number of Steps	Simple Random Walk	Lazy Random Walk
t=2	Q= 0.230428	Q= 0.235216
t=3	Q= 0.223314	Q= 0.227334
t=4	Q= 0.235517	Q= 0.254840

Example 4.2: We consider an undirected network with 643 vertices. Then apply the Walktrap algorithm and our extended Walktrap algorithm with $\mu = 0.1$. Then we get the results as below.
We apply both algorithms with $t = 2$ and 4. For our algorithm, we take $\mu = 1$ and obtain the corresponding Modularity values in table II. Figure 4, illustrates the clustering results of both algorithms with $t = 4$.

Remark 4.3: In both of the above examples, we see that for the same number of steps t, the clustering result will be better when taking $\mu = 0.1$ than using $\mu = 0$ (Walktrap algorithm)

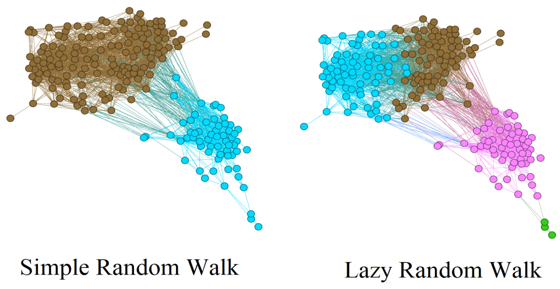

Fig. 3. Clustering the graph of 198 vertices with number steps is 4

TABLE II
Modularity value comparison table

Number of Steps	Simple Random Walk	Lazy Random Walk
t=2	Q= 0.361241	Q= 0.414941
t=3	Q= 0.357930	Q= 0.420204
t=4	Q= 0.426511	Q= 0.442939

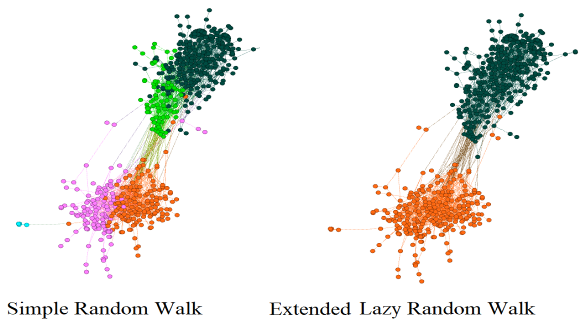

Fig. 4. Clustering the graph of 643 vertices with number steps is 4

V. Conclusion and further work

In this paper, we proposed an extended lazy random walk and used it to construct the distance between vertices on the graph. Since then, we have extended the Walktrap algorithm. We were also programmed to demonstrate the efficiency of our algorithm. In the future, we will study to extend this algorithm for directed graphs and, at the same time, research to improve the computational complexity of this algorithm.

Acknowledgments

This research was supported by the Institute of Mathematics, Vietnam Academy of Science and Technology, Project code: NVCC01.14/22-22.

References

[1] D. Aldous and J. A. Fill. Reversible Markov Chains and Random Walks on Graphs, chapter 2. Forthcoming book, http://www.stat.berkeley.edu/users/aldous/RWG/book.html.

[2] M. S. Aldenderfer and R. K. Blashfleld. Cluster Analysis. Number 07-044 in Sage University Paper Series on Quantitative Applications in the Social Sciences. Sage, Beverly Hills, 1984.

[3] J. Bagrow, E. Bollt. A local method for detecting communities. Physical Review E 72(4 Pt 2). 2005: 046108.

[4] A. Capoccia, V.D.P. Servedioa, G. Caldarella, F. Colaiori, Detecting communities in large networks, July 2005 Physica A: Statistical Mechanics and its Applications 352(2-4):669-676.

[5] L. da F. Costa. Hub-based community finding, arXiv:cond-mat/0405022, 2004.

[6] A. Clauset. Finding local community structure in networks. Physical Review E, 72:026132, 2005.

[7] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. Physical Review E, 70(6): 066111, 2004

[8] S. v. Dongen. Graph Clustering by Flow Simulation. Ph.D. thesis, University of Utrecht, May 2000.

[9] L. Lovász. Random walks on graphs: a survey. In Combinatorics, Paul Erdős is eighty, Vol. 2 (Keszthely, 1993), volume 2 of Bolyai Soc. Math. Stud., pages 353– 397. János Bolyai Math. Soc., Budapest, 1996.

[10] M. E. J. Newman. Fast algorithm for detecting community structure in networks. Physical Review E, 69(6):066133, 2004.

[11] B. S. Everitt, S. Landau, and M. Leese. Cluster Analysis. Hodder Arnold, London, 4th edition, 2001.

[12] A. Grover, J. Leskovec. node2vec: Scalable Feature Learning for Networks. KDD : Proceedings. International Conference on Knowledge Discovery Data Mining: 855-864, 2016.

[13] MC. Gonzalez, HJ. Herrmann, J. Kertesz, T. Vicsek. Community structure and ethnic preferences in school friendship networks. Physical A 379: 307-316, 2007.

[14] C. Moore. The Computer Science and Physics of Community Detection: Landscapes, Phase Transitions, and Hardness. Bull. EATCS 121, 2017.

[15] M. E. J. Newman. Fast algorithm for detecting community structure in networks. Physical Review E, 69(6):066133, 2004.

[16] P. Pons and M. Latapy. Computing communities in large networks using random walks, Journal of Graph Algorithms and Applications, volume 10, no. 2, 2006, Pages 191–218, 2006.

[17] J-F. Rural, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, et al. Towards a proteome-scale map of the human protein-protein interaction network. Nature 437: 1173. 2005.

[18] J. Reichardt, S. Bornholdt. Detecting fuzzy community structures in complex networks with a potts model. Physical Review Letters, 93:218701, 2004.

[19] U. Stelzl, U. Worm,M. Lalowski,C. Haenig, F. Brembeck F, et al. A Human Protein-Protein Interaction Network: A Resource for Annotating the Proteome. Cell 122: 957-968, 2005.

[20] H. Zhou and R. Lipowsky. Network Brownian Motion: A New Method to Measure Vertex-Vertex Proximity and to Identify Communities and Subcommunities. International Conference on Computational Science. 2004, 1062-1069.

[21] F. Wu, B. A. Huberman. Finding communities in linear time: A physics approach. The European Physical Journal B, 38:331 338, 2004.