只是对 girvan-newman algorithm 做了简略运算的操作，对结果的精度无调整

MIPRO 2014, 26-30 May 2014, Opatija, Croatia

Community structure in networks: Girvan-Newman algorithm improvement

Ljiljana Despalatović*, Tanja Vojković**, Damir Vukičević**
*The University Department of Professional Studies, University of Split, Croatia
**Faculty of Natural Sciences and Mathematics, University of Split, Croatia
ljiljana.despalatovic@oss.unist.hr

***Abstract*—Real world networks often have community structure. It is characteristic that the groups of nodes are connected denser within themselves and rarely with each other. The Girvan-Newman method for the detection and analysis of community structure is based on the iterative elimination of edges with the highest number of the shortest paths that go through them. By eliminating edges the network breaks down into smaller networks, i.e. communities. This paper introduces improved Girvan-Newman method where multi-edge removal is allowed, and presents the results of the application of both methods to the existing real social network (Zachary karate club), the computer-generated network and the tumor genes and their mutations network. The improved algorithm in practice reduces the number of operations, but retains the same computational complexity, so it cannot be applied to networks with a very large number of nodes. The most important feature of our improvement is that the result is graph-theoretical invariant, while original algorithm depends on the vertex labeling.**

***Index Terms*—Girvan-Newman, complex networks algorithms, edge betweenness**

I. Introduction

A network is a set of nodes and links connecting them. Studying network properties belongs into graph theory area which is a part of discrete mathematics. With the development of computers that can perform operations on very large amounts of data, modeling and analysis of network performance is extended to other scientific disciplines and becomes an interdisciplinary topic.

In mathematics, network is represented with a graph, where nodes are *vertices* and links *edges*. *Graph* is ordered pair $G = (V, E)$, where $\varnothing \neq V = V(G)$ is set of vertices, $E = E(G)$ is set of edges such that every $e \in E$ connects two vertices $u, v \in V$ called *endpoints* of e. Two connected vertices are *adjacent* vertices and we say that the edge and its endpoint are *incident*. A *loop* is an edge that connects a vertex to itself. The number of edges incident to a given vertex is the *degree* of that vertex, where every loop is counted twice. A graph is *regular* if all its vertices have the same degree.

In the real world, complex systems in different disciplines form networks such as social networks, World Wide Web, transport networks, citation network, metabolic networks, protein interaction networks, and gene regulatory network. Network exist in almost every area of human existence, from neuroscience to sociology, from economy to information technology. Those networks are called *complex networks* or *real-world networks*. Although graph theory is almost 300 years old (Euler solution for Könisberg bridges problem in 1736. is considered as the birth of the graph theory), the study of complex networks is a young discipline, only a little more than a decade old.

Unlike random (Erdos-Renyi model) and regular graphs, which were the focus of the graph theory before the Erdos-Renyi model, most real world networks are not random nor regular and have common topological properties. Some of them are small world effect, scale-free property, clustering or network transitivity, and community structure.

The property of complex networks that nodes are divided into groups or communities in which the links connecting nodes within the community are dense and the connections between nodes from different communities are sparse is the focus of this study. The possibility of detecting communities in networks can have practical applications. Communities in social networks could represent people with similar preferences or interests. The epidemiology could use the results to recognize the communities and the connections between people in order to prevent the spread of disease [1]. Kininmonth et al. [2] used the community detection methods to analyze genetic information exchange among the coral reefs through transport larvae. Communities on the Internet could be pages with similar content. From these examples, it is clear that the detection of communities in complex networks has broad application in a variety of scientific disciplines.

The rest of the paper is organized as follows. In Section 2 we present methods used for community structure detection in complex networks. Section 3 describes the Girvan-Newman method and our improvement of the method. In Section 4 we introduce modularity, a qualitative measure of network decomposition. Section 5 presents and discusses results achieved with our modification, and finally, Section 6 concludes the article.

II. Methods for community structure detection in complex networks

Main methods for finding communities in a network are variants of hierarchical clustering methods used in social network analysis [3]. Hierarchical clustering could be agglomerative or divisive, depending on decision to start from empty network and add the edges to form communities, or to start with complete network and remove edges until the communities are formed. In both cases, the process could be represented with *dendrogram*, which is a hierarchical tree with clusters as a nodes in tree, and single nodes as leaves. In agglomerative process, dendrogram is built from leaves to the root, and in divisible process from root to the leaves (Fig. 3a).

A. *Agglomerative methods.*

The traditional hierarchical agglomerative clustering algorithm starts with empty graph which consist of nodes of original graph without edges. In every step of the algorithm edges are added, starting from "stronger" to "weaker" links.

Edge weight can be calculated in different ways. For example, edge weight could be the number of node-independent or edge-independent paths between vertices. Two paths are node-independent if they share no other vertices than the path endpoints. Similarily, they are edge-independent if they share no edges. The number of those paths represents the number of vertices (or edges) which should be removed from the graph in order to disconnect path endpoints [4].

New communities are formed in the consecutive steps of the algorithm. Iterative algorithm process could be stopped at any iteration, but usually, it is done until all the edges are added to the graph. In the end, all the nodes and edges form one community.

B. *Divisive methods.*

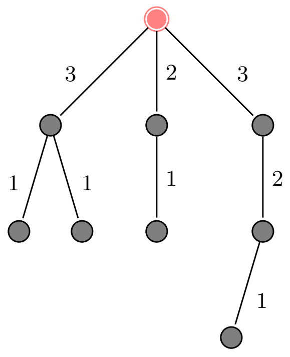
Opposite of agglomerative methods, divisive methods start from the complete graph and, in every iteration, remove the edges with the highest weight. In every step weight calculation is repeated, since the weight of remaining edges changes with edge removal. The Girvan-Newman algorithm, which will be described in detail in the next section, is an example of divisive algorithm. In that algorithm, the order of removal of the edges with the highest weight is not defined, so it could produce different results depending on implementation. Modifications of that algorithm exist in which the community structure is built for every possible order of highest weight edge removal.

III. GIRVAN-NEWMAN METHOD

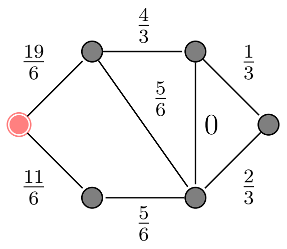
Girvan-Newman method is divisive method where edge weight is the number of shortest paths passing through the edge. That value is called *edge betweenness* and it is a generalization of central vertex betweenness which determines vertex influence on other vertices in network. Vertex betweennes is the number of shortest paths passing through the vertex, therefore, edge betweenness is the number of shortest paths passing through the endpoints of the edge.

We can describe Girvan-Newman algorithm in the following way:

1) Calculate edge betweenness for every edge in the graph.



(a) A tree



(b) Not a tree

Figure 1: The number of shortest paths in a graph: a) In a tree there is unique path from source to every vertex. b) If graph is not a tree there is more than one path to every vertex. Each edge has weight which is fraction of the paths flow through the edge.

2) Remove the edge with highest edge betweenness.
3) Calculate edge betweenness for remaining edges.
4) Repeat steps 2-4 until all edges are removed.

In order to calculate edge betweenness it is necessary to find all shortest paths in the graph. The algorithm starts with one vertex, calculates edge weights for paths going through that vertex, and then repeats it for every vertex in the graph and sums the weights for every edge.

Let us choose source vertex $s \in V$ in a graph. In the most simple case graph is a tree (Fig. 1a) and there exists only one shortest path from the source vertex to any other vertex. Starting from the leaves we assign the value 1 to the edges that connect leaves with the rest of the tree (because there is only one shortest path to s passing through that edge). Moving upwards in the tree we assign edge value as a sum of values assigned to the edges directly below it, increased by 1. The number of shortest paths in the tree from source vertex to every other vertex passing through particular edge is that edge weight value. By repeating the process for every vertex and calculating the sum of weight values for every edge, we calculate edge betweenness for every edge.

If a graph is not a tree (Fig. 1b), it is possible that more than one shortest path connects source vertex with some other vertex. In that case, value $\frac{k}{l}$ is assigned to the edge on shortest path to source vertex, where k is the number of shortest paths to source vertex from the endpoint of the edge that is closer to source and l is the number of shortest paths from source vertex to other endpoint of the edge, and multiplied by the number of shortest paths from source that pass through edges below farther vertex incresed by one. Assignment starts from the edge that has maximum distance from source vertex.

The algorithm for calculating edge betweenness is performed in two parts. In the first part of the algorithm, using breadth-first search, distance from source is assigned for every vertex and also the number of shortest paths from source to vertices. In the second part, starting from edge incident to the vertex with maximum distance from the source vertex as endpoint (the last vertex visited in the first part on the algorithm), the numbers of shortest paths passing through edges are calculated for every edge. For every vertex $i \in V$ the triple (d_i, w_i, b_i) is calculated, where d_i is the distance from the source vertex, w_i is the number of shortest paths from source vertex to vertex i, and b_i is the number of shortest paths between source vertex to any vertex in graph that pass through vertex i.

For this algorithm we denote $Adj(v)$ as the set of all vertices adjacent to $v \in V$.

The first part of the algorithm for vertex marking:

```
1) For initial vertex $s \in V$ let $d_s = 0$, $w_s = 1$, $b_i = 0$.
2) Let $d_v = \inf$, $w_v = 0$, $b_v = 1$ for all $v \neq s \in V$.
3) Create queue $Q$, $Q \leftarrow \{s\}$. Create list $L$, $L \leftarrow \{s\}$.
4) While Q is not empty:
    a) Dequeue $i \leftarrow Q$.
    b) For each vertex $j \in Adj(i)$:
        i) If $d_j = \inf$ then $d_j = d_i + 1$, $w_j = w_i$. Enqueue $j \rightarrow Q$. Push $j \rightarrow L$.
        ii) If $d_j \neq \inf$ and $d_j = d_i + 1$ then $w_j += w_i$.
        iii) If $d_j \neq \inf$ and $d_j < d_i + 1$, do nothing.
```

Efficient implementation of this part of the algorithm could be done by using abstract data type queue.

The second part of the algorithm starts from the vertex that was last marked in the first part of the algorithm and visits vertices in reverse order than they were visited in the first part of the algorithm. Only one shortest path from source passes through the last marked vertex.

The second part of the algorithm for edge betweenness calculation:

```
1) While $L$ is not empty:
    a) Pop $i \leftarrow L$.
    b) For each vertex $j \in Adj(i)$:
        i) If $d_i < d_j$ then $b_i = 1 + \sum_j \sigma_{ij}$.
        ii) If $d_i > d_j$ then $\sigma_{ij} = \frac{w_j}{w_i} * b_i$.
```

Both parts of the algorithm are performed for all source vertices s and edge betweeness for every edge is calculated as a sum of the edge betweennesses calculated in every step.

The computational complexity of this part of algorithm is $O(mn)$, where m is the number of edges and n is the number of vertices. After each edge betweenness calculation, the edge with highest edge betweenness is removed and the algorithm is repeated until there is no remaining edge. The algorithm complexity is therefore $O(m^2n)$.

In complex networks it is often the case that more edges have the same highest edge betweenness. Since the recalculation of edge betweenness has $O(mn)$ complexity, in order to reduce the number of calculations we could remove all edges with the highest edge betweenness in the same step. On the other side, Girvan-Newman algorithm does not say anything about the order of removal for those edges leaving that detail to implementation.

Consequently, we introduce modification of the Girvan-Newman algorithm:

1) Calculate edge betweenness for every edge in the graph.
2) Remove all edges with highest edge betweenness.
3) Recalculate edge betweennes for remaining edges.
4) Repeat 2-4 until graph becames empty.

Worst-case time complexity is still $O(m^2n)$, but in networks with strong community structure the number of calculations could be significantly reduced.

The main problem of Girvan-Newman algorithm is the fact that it is not really an algorithm that has a graph as input and community structure as output. Namely, if the labeling of the vertices of the graph is rearranged, than the result of Girvan-Newman algorithm may change. Hence, there is no unique output for a given input (which should be a property of any algorithm). Our modification amends this shortcoming of the Girvan-Newman algorithm. Namely, our community structure indeed does not depend on the labeling of the vertices of the graph.

Also, this modification may reduce the number of operations of the algorithm. However, this strongly depends on the observed graph. There are graphs for which no improvement will be obtained and there are graphs for which significant improvement will be obtained. The most drastic example may be edge-transitive graphs (i.e. graphs such that for each two edges e and f there is the automorphism that maps e to f). Standard Girvan-Newman algorithm has complexity $O(m^2n)$ on this graphs also, while our algorithm does not repeat calculation for each edge, but does everything in a single step reducing complexity to $O(mn)$.

IV. Modularity

The algorithms used for community structure detection in networks give an overview of possible communities. Simulation of decomposition done by Girvan-Newman method gives more information of network structure. Knowing network structure, it is possible to predict critical connections in the network, and therefore, control the network. For example, in the electricity distribution network detecting critical connections could be a method to prevent the collapse of the system that has occurred in the United States in 1965, 1996, and in September 2011.

However, the question is when an optimal decomposition of a network into communities is reached. To answer that question it is not necessary to decompose network in communities of the size one. Qualitative measure of network decomposition is called **modularity** [5].

For a particular partition of a network into k communities, let us define $k \times k$ matrix E such that e_{ij} is the fraction of number of edges that connect vertices between community i to community j in the total edge number. Then on the diagonal of the matrix E is the fraction of edges that are located within the same community, so the trace of the matrix $tr(E)$ is the fraction of edges that will not be removed in process of removing edges. A good partition is the partition with high value of the trace. However, that number will be the highest in the case where all the vertices are in the same community, so it does not always give necessary information about the structure of the communities.

We define modularity:

$$Q = \sum_i (e_{ii} - a_i)^2 = tr(e) - \|E^2\|,$$

where $\|\|$ is the sum of matrix elements, as the fraction of edges that connect vertices within communities decreased with the expected fraction of edges within partition in random graph [6]. When the fraction of edges within the communities is higher than it is in random graph, the value is $Q = 0$. As Q is approaching the value 1, the community structure in network is better. In most cases, the value of Q is between 0.3 and 0.7 [5].

Value Q is calculated in every step of divisible algorithms. The maximum value q gives us the best partition of the graph.

V. Results

We tested the Girvan-Newman algorithm with the single edge removal and modified algorithm with the multiple edges removal on three different complex networks. The first, Zachary's karate club; the second, computer generated random network; and the third, network of cancer genes with co-occurring and anti-co-occurring mutations.

A. Zachary's Karate Club

Zachary's karate club is the example of the real-world network for which it is known how the network was split in reality. Fig. 2 shows the friendship network between 34 members of a karate club at a US university observed by an anthropologist Wayne Zachary for a period of three years, from 1970 to 1972 [7].

At the beginning of the study there was a conflict between the club president (node 34) and the karate instructor (node 1). During the study club was divided into two parts and has become the most frequently tested dataset in the network communities detection area. The Girvan-Newman algorithm divides the network into two communities with an accuracy of 97% (which is also Zachary's accuracy) with wrong classification of the node 3 (in Zachary's study the node 9 was wrongly

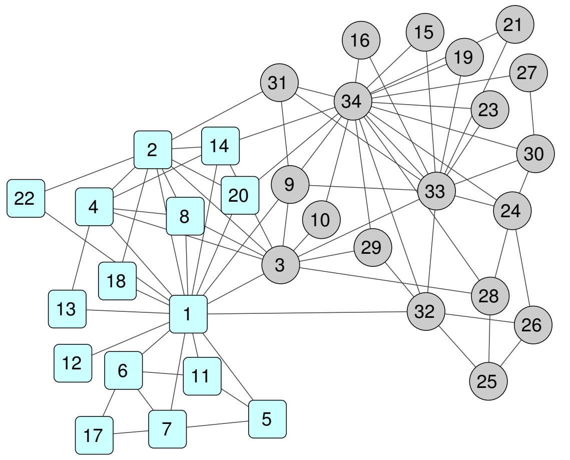


Figure 2: Zachary karate klub

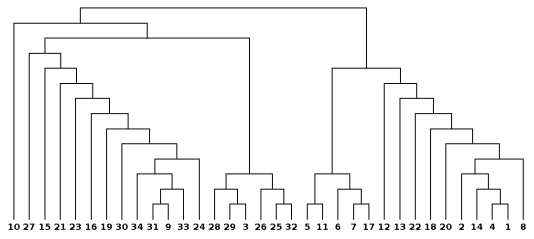
classified). Fig. 3a shows the dendrogram generated using the Girvan-Newman algorithm, while Fig. 3b shows the dendrogram generated using implementation of modified method with multiple edge removal. It is clear that our method leads to faster separation of the communities i.e. reduced number of operations (Fig. 4). The algorithm modularity values are high when the network is divided into two communities which is expected concerning the nature of community and the process of the separation. However, modularity is the largest in the network partition to the three communities. On Fig. 2 we can see that there exists a community of five nodes (5, 6, 7, 11, 17) with high number of links within community and small number of links to other communities.

B. Computer generated network

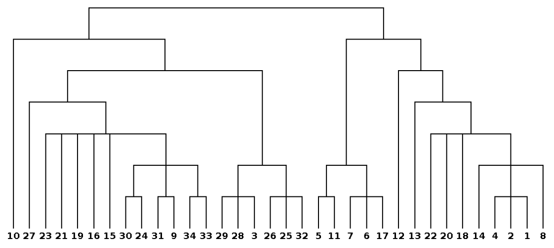
In addition to real-world examples we analysed computer-generated network with strong community structure. For this purpose, network with 48 vertices divided into communities of 16 vertices is generated with probability p_{in} that edge is within community and p_{out} that edge is between communities. Fig. 5 presents the modularity values for Girvan-Newman algorithm and for our multiple-edge-removal modification of the algorithm. The modularity function maximum is reached in both algorithms when the network is divided in three communities, but needed fewer steps to complete calculation in the latter case. Both algorithms correctly classify all nodes.

C. The network of cancer genes with co-occurring and anti-co-occurring mutations

The network of cancer genes with co-occurring and anti-co-occurring mutations (CCA network) is presented in [8]. CCA networks are studied in order to better understand the development of cancer and community detection in such networks is an important property. The cancer genes are nodes and co-occurring and anti-co-occurring mutations are links between them. The tested network has 306 nodes and 1366

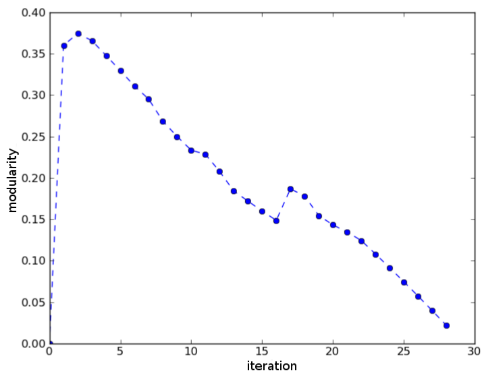


(a) Single edge removal

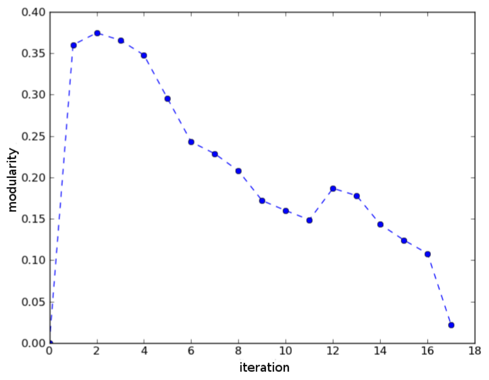


(b) Multiple edge removal

Figure 3: Zachary's karate club

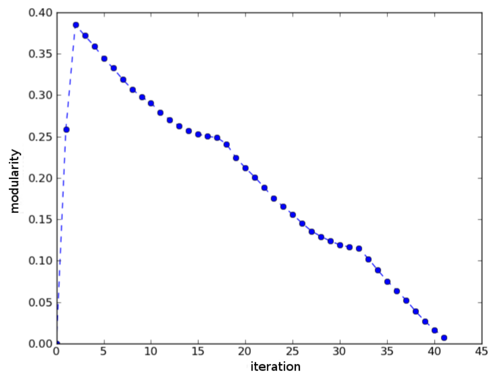


(a) Single edge removal

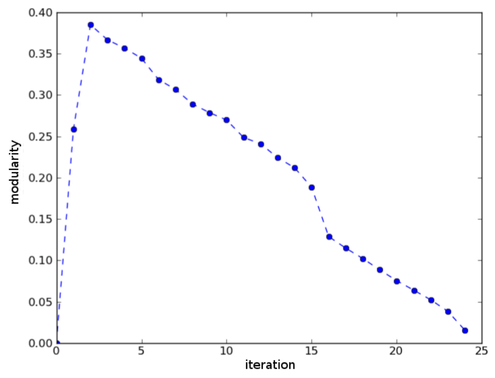


(b) Multiple edge removal

Figure 4: Zachary's karate club - modularity



(a) Single edge removal



(b) Multiple edge removal

Figure 5: Computer generated network modularity

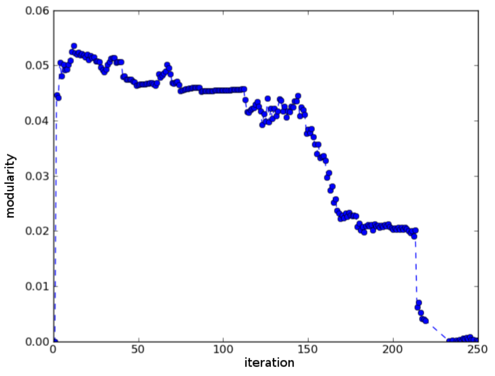
edges, of which 1355 are co-occuring and 11 are anti-co-occuring.

The number of iterations in this example needed to divide the network into communities with modularity 0 is much smaller when multiple edge removal is done. Both algorithms correctly classify all the nodes.

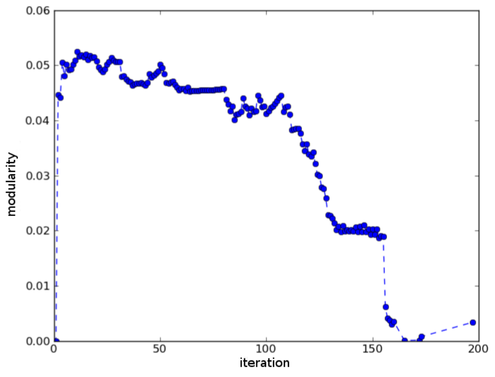
VI. Conclusion

Community detection in networks is important for different scientific and practical areas. It is not a surprise that in this article we refer to articles of physicists, sociologists, mathematicians, biologists, epidemiologists, etc. Complex network research is young discipline and with its rapid development reflects the flourishing of interdisciplinary research. Graph theory is widely used in various applications in different areas. Some of them are mentioned in this article. The robustness of a network and analysis of system collapse due to the elimination of nodes or links between nodes is of great importance in the study of networks. How can we stop the propagation of diseases? How can we determine the most important airlines or highways? Algorithms decribed in this article help us to experience the world around us.

Girvan-Newman algorithm is one of the first algorithms that deals with detecting communities in networks and as such suffers from certain ”childhood diseases”. Number of operations is proportional to m^2n, or n^3 for sparse networks, which limits performance on networks with up to tens of



(a) Single edge removal



(b) Multiple edge removal

Figure 6: A network of cancer genes with co-occurring and anti-co-occurring mutations modularity

thousands of nodes. Methods based on modularity, [6] [9] [10], the use of the eigenvectors of matrices [11], and greedy algorithms [12] give us smaller complexity than Girvan-Newman method. Regardless, in this article we focused on the Girvan-Newman method and its improvement, as a starting point for any subsequent methods. This small algorithm improvement reduces the number of operations, but still not enough to apply it on large complex networks. But on smaller networks it has a high percentage of correct classification of nodes and it is independent of vertex labeling, which puts it ahead of other methods.

References

[1] S. Kitchovitch and P. Liò, "Community Structure in Social Networks: Applications for Epidemiological Modelling," *PloS one*, vol. 6, no. 7, p. e22220, 2011.

[2] S. Kininmonth, M. J. H. Van Oppen, and H. P. Possingham, "Determining the community structure of the coral seriatopora hystrix from hydrodynamic and genetic networks," *Ecological Modelling*, vol. 221, no. 24, pp. 2870–2880, 2010.

[3] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. Hwang, "Complex networks: Structure and dynamics," *Physics Reports*, vol. 424, no. 4-5, pp. 175–308, 2006.

[4] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, 2002.

[5] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, vol. 69, no. 2 Pt 2, p. 16, 2004.

[6] A. Clauset, M. E. J. N. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, pp. 1–6, 2004.

[7] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, no. 4, pp. 452–473, 1977.

[8] Q. Cui, "A network of cancer genes with co-occurring and anti-co-occurring mutations," *PLoS ONE*, vol. 5, no. 10, p. 8, 2010.

[9] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, no. 2, pp. 1–5, 2004.

[10] D. Chen, Y. Fu, and M. Shang, "A fast and efficient heuristic algorithm for detecting community structures in complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 13, pp. 2741–2749, 2009.

[11] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, vol. 74, p. 036104, Sep 2006.

[12] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008.