Fast Unfolding of Communities in Large Networks Using Louvain Community Detection Algorithm

Diksha Prakash

Spring - 2019

Networks provide a convenient way to represent complex systems consisting interacting entities. Many networks contain "communities" of nodes that are more densely connected to each other than to nodes in the rest of the network. This term paper discusses a simple method to extract the community structure of large networks. The method is a heuristic method which is based on 'Modularity Optimization'. The method is claimed to outperform majority of the other known community detection method in terms of computation time. Moreover, the overall quality of the detected communities is quite impressive, which is quantified by the 'modularity'. An interesting application of the same is based on identifying language communities in a Belgian mobile phone network of approximately 2.6 million customers. The application is also extended to ad-hoc modular networks.
Keywords: *Network, Communities, Modularity, Modularity Optimization, Community Detection*

1 Introduction

1.1 Background:

Communities refer to groups of nodes within a given network which are more densely connected to each other as compared to other nodes. Complex networks have a common feature wherein they consist of multiple sets of nodes which tend to interact more with one another than with those outside of the set. For instance, social networks might consist of closely knit communities of friends with occasional friendship ties between different and distant communities. Suppose one maps a network of projects going on in a large company. It could be noticed that certain projects will likely have more conceptual overlap and mutual dependency than the other projects. Another example is the protein interaction networks where certain groups of proteins interact with each other more frequently than with proteins in

other groups.
Back in 1962, it was proposed by H.A. Simon that the type of community structure might be a defining characteristic of the complex systems, those in which many constituent and interacting elements organize adaptively to achieve any higher-order function. The reasoning for the same was that individual participants have a much higher chance of collectively achieving the higher-order function if that particular function could be iteratively achieved by constructing intermediate stable forms which in turn achieve simpler functions. These intermediate stable forms are referred to as communities or modules. The first-order intermediate forms can be described as the communities in terms of the original nodes. But, the interactions between these first-order communities can generate second-order communities which somewhat accomplish further complicated functions, and this goes on. In this way, these hierarchical structures can result into 'complex adaptive systems' [1]. However, community structure is a consistent feature of complex networks even when the idea of achieving some higher-order function isn't considered.
In general, technological, social and information systems can be depicted as complex networks consisting of interconnected nodes; testifying both organization and randomness [2, 3].

1.2 Motivation:

In general, the typical size of mobile phone networks, social networks etc typically count in millions and billions and hence a novel method to retrieve comprehensive information from the networks of this scale is highly desirable. One viable option for the same is to decompose and aggregate the networks into communities which have a high level of co-relation between the inter connected nodes [4]. When we have a a network described by a data set, the underlying community structure is typically unknown to us. Often even the number of first-order communities we should be looking for is unknown to us, let alone the number of hierarchical levels of communities that should be considered. Meanwhile, the community and hierarchical structures are often of prime importance for understanding the overall functioning of the provided complex system. This echoes the need for a good and reliable community detection algorithm. On the personal front, this was also an interesting topic to me as in the past, I have constantly heard of it while dealing with and optimizing the Cypher queries for Neo4j (Graph Database). This gave me an opportunity to dig deeper into what was previously not very clear and understandable to me.

1.3 Overview:

The community detection algorithms partition a network into multiple communities consisting of densely connected nodes, wherein the nodes belonging to distinct communities are sparsely connected. Till date, many algorithms have been proposed to achieve the above mentioned partition in a fast and reliable manner. In the recent years, with ever growing network data sets and their inadmissible impact on daily lives, this topic has been a hot bed for research. Community detection algorithms cab be divisive, agglomerative or optimization based. Divisive algorithms detect inter-community links and remove them from the network [5, 6, 7]. Agglomerative algorithms merge similar communities or nodes recursively [8]. Whereas, the optimization methods aim for the maximisation of the objective function involved [9, 10, 11]. 'Modularity' refers to the metric which quantifies the quality of partitions formed as a result of these community detection algorithms. It was proposed by Newman and Girvan [3] and is a scalar value between -1 and 1 and is based on measuring the the density of links inside communities as compared to links between communities [3, 12].

$$Q = \Sigma_c[B_{cc} - W_{c^2}] \tag{1}$$

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

Mathematically, it can be defined as equation 1 where $B_{c_1\,c_2}$ denotes the fraction of all edge weights in the network that connect community 'c1' to community 'c2'. W_{c1} is defined as the total edge weight penetrating community 'c1' i.e. $W_{c_1} = \Sigma_{c_2}(B_{c_1\,c_2})$ Equation 1 describes the condensed version of the modularity index. The actual equation described in [3] is equation 2 where, A_{ij} represents the weight of the edge between i and j, $k_i = \Sigma_j A_{ij}$ represents the sum of the weights of the edges attached to the vertex i, c_i is the community to which vertex 'i' is assigned and the δ-function $\delta(u, v)$ is 1 if u=v and 0 if $u \neq v$. Mathematically, $m = \frac{1}{2}\Sigma_{ij}A_{ij}$.

According to the definition above, the problem of modularity optimization can be framed as assigning communities in a manner that the sum of the elements that contribute to the community are as positive as possible (From the equation 2, it can be noticed that the δ function kills those terms in the sum which correspond to pairs of nodes belonging to different communities i.e. $u \neq v$). The underlying problem is that we can have cases where putting nodes 1 and 2 in the same community makes sense and putting nodes 2 and 3 in a community also makes sense. But it might be possible that $A_{13} - \frac{k_1 k_3}{2m}$ is very negative, hinting that nodes 1 and 3 shouldn't

be in the same community. This is a basic example of the kind of 'frustration' which makes modularity optimization problem really complex. Thus, exact modularity optimization is a problem that is computationally hard and hence there's a need for approximation algorithms when dealing with large networks. The fastest known optimization algorithm is based on recurrently merging communities that optimize the product of modularity [9]. This algorithm follows a divisive approach and is greedy in nature and can end up producing quite low values of modularity. Moreover, the method has an inclination to produce super-communities that contain large fraction of nodes, even in the case of synthetic networks which are devoid of any significant community structure.

2 Louvain Community Detection Algorithm

For the purpose of illustration of Louvain Method, a 'Connected Caveman Graph' [13] will be used in this term paper. The (connected) caveman graph is a graph born out of social network theory and is formed by modifying a set of isolated k-caves. This modification is done by removing one edge from each cave and using that edge to connect to a neighboring cave along a central cycle such that all the 'n' caves form an unbroken single loop [13]. In these types of network, we begin

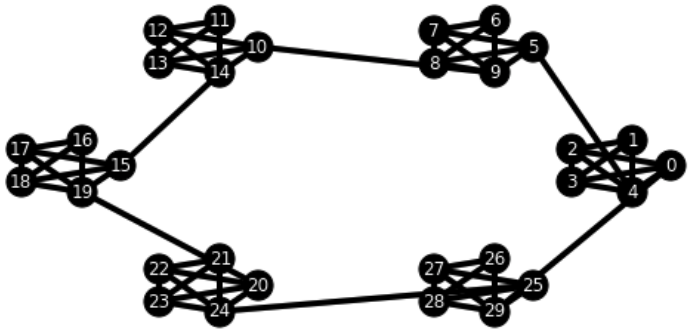

Figure 1: Connected Caveman Graph

with N_{cl} fully-connected cliques (caves) of M nodes each. Next, these cliques are arranged in a circle. Then, one random node is taken from each clique and rewired such that the clique is now connected to its nearest neighbor in the clockwise direction. This is done once for each clique and finally we end up with something like Figure 1. It is assumed that all the links in this initial network have been assigned unit weight. The Louvain Method described below, uses this test network. Here, the 'intuitive' partition consists of the six distinct communities consisting

five nodes each that have been assigned by hand.
At the start of the Louvain Method, each node is assigned to its own community, thus, in the connected caveman network above we have 30 initial communities with each containing a single node. The Stage 1 of the Louvain Method can be tagged as 'Community Reassignments'.

2.1 Stage 1 : Community Reassignments

Iteration is performed through all the nodes in the network. The change in modularity upon replacing the node from its current community to the community of one of its neighbors is computed. The modularity change for this replacement in each of the node's neighbors is calculated. If these modularity changes are negative, the node is left in its current community. When the modularity changes is calculated to be positive, the node is moved to the community which gives the most positive change for modularity. In case of a tie, arbitrary resolutions happen. This procedure is repeated for each node until one pass through every nodes results in no community assignment changes.
Let us consider the connected caveman graph 1, the few steps of this community reassignment can be analyzed through change by hand. Let us assume that in the current implementation of the Louvain Method, the first node considered was 29, which is in turn connected to the nodes 25-28. If nodes 25 and 29 were to be merged in the same community, it would mean activating the terms $A_{25,29} - \frac{k_{25}k_{29}}{2m}$ and $A_{29,25} - \frac{k_{25}k_{29}}{2m}$ in equation 2. Each of these terms will have a value $1 - \frac{25}{120} = \frac{19}{24}$. Thus, it can be inferred that by performing the community reassignment, the modularity will be amplified by $\frac{19}{12}$. For the given scenario, co-incidentally, all of the possible mergers for node 29 tend to have this same modularity amplification. Thus, any one of them can be accepted here. Let us assume that the implementation assigns node 29 to the same community as node 25. The method then moves ahead to consider another randomly selected node.
Co-incidentally, for the connected caveman graph 1, merging any pair of neighboring nodes will amplify the modularity by the same amount as calculated above i.e. $\frac{19}{12}$. In particular, this holds even for distinct pairs say, nodes 8 and 10, which clearly belong to different communities. This implies that, the initial stage of the community reassignment, pairs of distinct nodes have a likelihood of being tagged to the same community. However, by exercising certain precautions, mistakes like this can be re-mediated while the community reassignment stage progresses.
Post the first community assignment stage, the community assignments as in Figure 2 can be found using the current implementation of the Louvain Method.

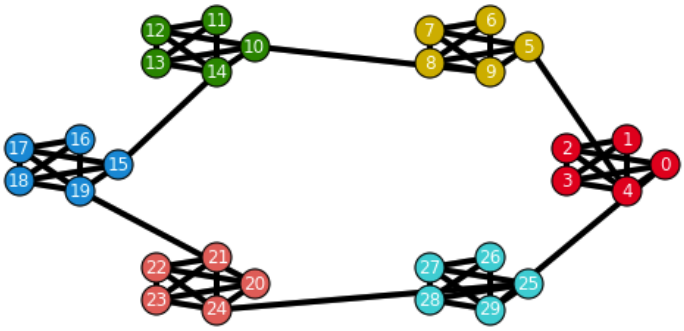

Figure 2: Community Assignment Stage Result

2.2 Stage 2: Coarse Graining

This stage in the Louvain Method uses the communities that were discovered in the former stage i.e. 2.1 to define and demonstrate a coarse-grained new network. In this network, the newly discovered communities (the end result of Stage 1) are treated as nodes. The weight of the edge between these nodes, which represent two communities, is essentially the sum of the weights of the edges between the lower-level constituent nodes present in each individual community. Self-loops are generated by the links within each community in this coarse-grained new network. In the simple connected caveman network 1 under consideration, there exists only one link of unit-weight which connects the neighboring communities. Thus, intuitively, the links between these new coarse-grained communities have unit weight too. If there existed multiple links between the communities, the coarse-grained link's weight would be equal to the summation of all the lower-level links. Here, within each community there exist $5421 = 9$ unit-weight links. Thus, it can be said that the self-loops have a weight 9. The coarse-grained network obtained will look like Figure 3 where edge weights are denoted by the black numbers present with the links . The labels for the communities are denoted by the white numbers on the nodes. (NOTE: For all the analysis results above, the python package 'NetworkX' has been used.

2.3 Stage 3: Repeated Iteration of Stages 1 and 2

The Stage 3 of the Louvain Method constitutes of the repeated iteration of stages 1 2.1and 22.2. Application of Stage 1 i.e. 'Community Reassignment Phase' to the coarse-grained graph, helps us find a second level of communities of communities of the nodes. Further, in the next usage of Stage 2, a new coarse-grained graph

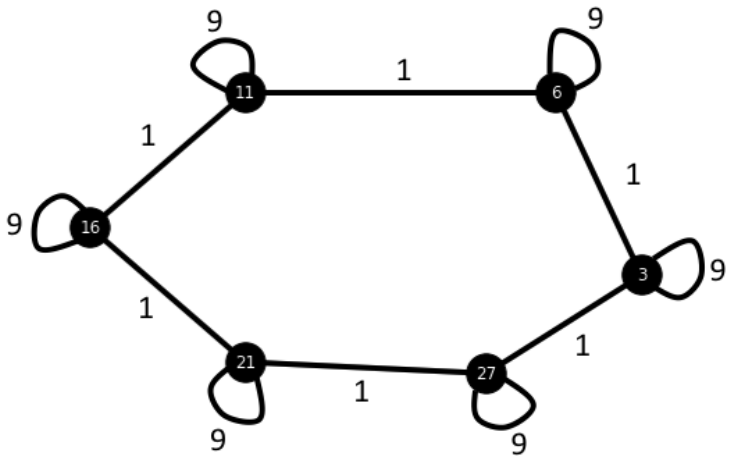

Figure 3: Coarse Grained Network

is defined at this comparatively higher level of the present hierarchy. This is continued till the outcome of stage 1 is no reassignments. This point on-wards, the repeated application of the stages won't ever result in any more modularity amplification or optimization. Hence the process gets completed.
For the example considered here i.e. Figure 1, the termination of the process happens on the second 'Community Reassignment' i.e. 2.1 phase. It is clear that if a merger of two communities is proposed in the coarse-grained graph i.e. Figure 3 above, it would result in a negative modularity change of $2 \times [120 \times 20120] = 143$. Thus the proposed merger will be accepted.

2.4 Experimental Results

In the original paper on the Louvain Method [14], the authors have tested their proposed algorithm against some of the existing other modularity optimization algorithms. The table excerpt from the paper indicating the comparatively faster nature of the algorithm for achieving similar modularities is shown in Figure 4[14].

	Karate	Arxiv	Internet	Web nd.edu	Phone	Web uk-2005	Web WebBase 2001
Nodes/links	34/77	9k/24k	70k/351k	325k/1M	2.6M/6.3M	39M/783M	118M/1B
CNM	.38/0s	.772/3.6s	.692/799s	.927/5034s	-/-	-/-	-/-
PL	.42/0s	.757/3.3s	.729/575s	.895/6666s	-/-	-/-	-/-
WT	.42/0s	.761/0.7s	.667/62s	.898/248s	.56/464s	-/-	-/-
Our algorithm	.42/0s	.813/0s	.781/1s	.935/3s	.769/134s	.979/738s	.984/152mn

Figure 4: Performance Comparison of Louvain Algorithm with existing algorithms

3 Implementation

The following module implements community detection and is based on Louvain Algorithm described in the Fast unfolding of communities in large networks [14]. The module relies on NetworkX [http://networkx.lanl.gov/] to handle the graph operations. The python implementation is as below [15].

```
import community
import networkx as nx
import matplotlib.pyplot as plt

# The following can be replaced personalized
# NetworkX graph loading depending on the format
G = nx.erdos_renyi_graph(30, 0.05)

#The best partition is first computed
partition = community.best_partition(G)

#Drawing and plotiing the results
size = float(len(set(partition.values())))
pos = nx.spring_layout(G)
count = 0.
for com in set(partition.values()) :
    count = count + 1.
    list_nodes = [nodes for nodes in partition.keys()
                if partition[nodes] == com]
    nx.draw_networkx_nodes(G, pos, list_nodes,
    node_size = 20, node_color = str(count / size))

nx.draw_networkx_edges(G, pos, alpha=0.5)
plt.show()
```

4 Real-World Applications

The original Louvain Method paper [14] reports an application wherein a study of a large Belgian phone call network was done. Here, the nodes depicted the customers

and the weighted links depicted the number of phone calls, over a six-month period, between any two customers. The example had a staggering 2.6 million nodes i.e. customers and 6.3 million links between them. The application of Louvain Algorithm suggested a hierarchy consisting six levels of communities. The top level of this hierarchy contained the communities strongly segregated by primary language and represented around 10,000 and more customers. All of these communities, except one, had an 85% or greater French or Dutch speaking majority. The only community having a more equitable distribution was placed at the interface between Dutch and French clusters in the top most level of the coarse-grained network. The authors [14] concluded the following about this community: "These groups of people, where language ceases to be a discriminating factor, might possibly play a crucial role for the integration of the country and for the emergence of consensus between the communities. One may indeed wonder what would happen if the community at the interface between the two language clusters... was to be removed."[14]
In Figure 5, around 2 million customers have been represented on this network. Here, The size of any node is proportional to the number of individuals present in the corresponding community and the colour coding on a red-green scale depicts the main spoken language in that community. The legend for the same is that red is representative of French and green is representative of Dutch. Here, the communities having more than 100 member nodes have only been plotted. There also exist intermediate communities having mixed colours between the clusters of the two main languages. A higher resolution zoom indicates that several sub-communities with less apparent language separation also thrive.
The Louvain Method finds a vast multitude of applications aimed at analyzing the real-world networks and data. Several of these hav been mentioned on the "official website" of the method `https://perso.uclouvain.be/vincent.blondel/research/louvain.html`
A few miscellaneous applications can be listed as under:

- Citation Networks: Analysis of collaboration communities
- Retail Transactions: Analysis of a network
- Social Networks: Analysis of online social networks like Twitter, LinkedIn, Flickr, Youtube, and LiveJournal

While going through the research papers available online, the topic which intrigued my interest is the *study of brain networks using the Louvain Method* [16]. Although the topic seemed very interesting, I wasn't able to understand the intricate details owing to the jargon. The authors of that study have found similar commu-

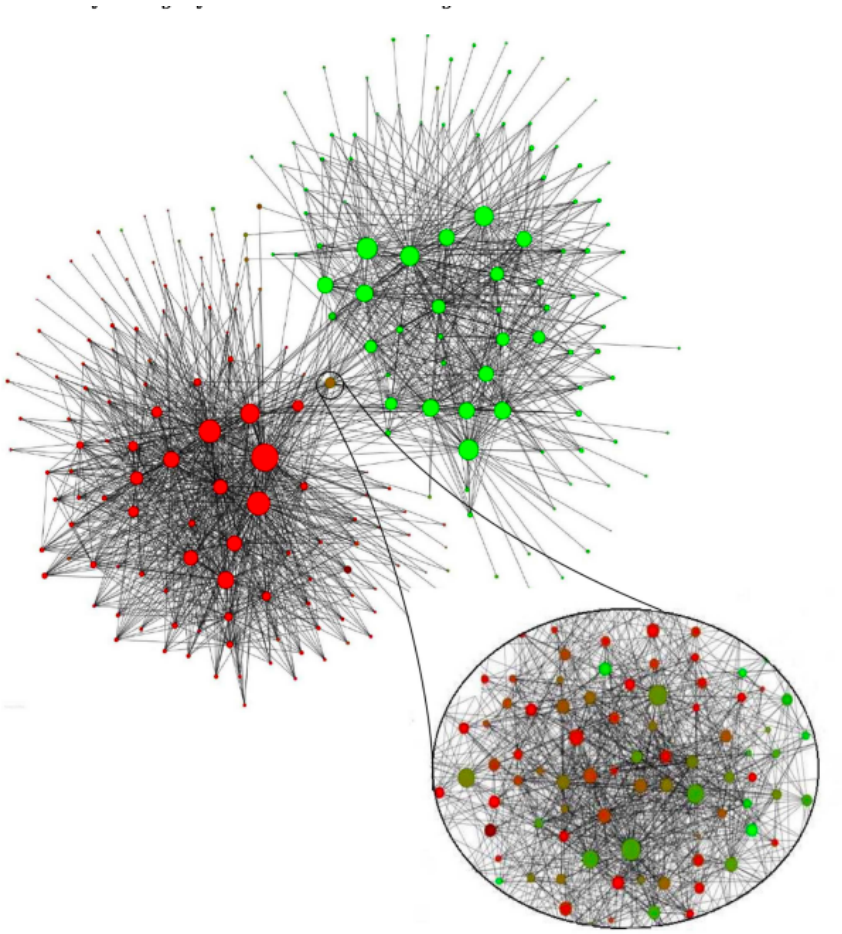

Figure 5: Graphical representation of the network of communities extracted from a Belgian mobile phone network [14]

nity structure across the human brains of 18 different use cases and modules that makes sense from a functional perspective.

5 Possible Pitfalls of Modularity Maximization

The Louvain Method along with the modularity optimization algorithms have found wide application across sundry domains. However, this doesn't stop the fundamental problems with these algorithms to be identified. A quick summary of the two of them are:

- **Resolution limit:** For larger networks, the Louvain Method doesn't stop with the "intuitive" communities. Instead, there's exists a second pass through the two stages namely, 'community modification' and 'coarse-graining', which leads to several of the intuitive communities to be merged together. Unfortunately, this is a general problem with modularity optimization algorithms themselves. They have issues in large networks while detecting small communities. It's a trait of the Louvain Algorithm that anything close to the 'intuitive' community structure is used as an intermediate step in the process.

- **Degeneracy problem:** There typically exist a profoundly large (in terms of network size) number of possible community assignments with modularity close to the maxima. This can pose to be a severe threat because, in the presence of a large number of high modularity solutions, it's

  - Difficult to find the global maxima
  - Difficult determination if the global maxima is actually more important scientifically than the local maxima that is capable of achieving the similar modularity.
  - It was shown by Good et al. that the different optimal (local) community assignments ere capable of possessing quite distinct structural properties [17].

  Such problems have been studied and discussed at length in [17]. The authors of the paper have concluded with the cautionary note about modularity maximization as follows:

  "...modules identified through modularity maximization should be treated with caution in all but the most straightforward cases. That is, if the network is relatively small and contains only a few non-hierarchical and non-overlapping modular structures, the degeneracy problem is less severe and modularity maximization methods are likely to perform well. In other cases, modularity maximization can only provide a rough sketch of some parts of a network's modular organization." [17]

6 Summary

Communities refer to groups of nodes within a given network which tend to be more densely connected to one another as compared to others. Modularity is a metric which quantifies the quality of the assignment of nodes to a community by

evaluating on the principle of 'on an average, how much more densely connected the nodes within a community are compared to how connected they would be in a given random network'. The 'Louvain Method' is an algorithm for detecting communities in networks and advocates maximizing the modularity. The algorithm consists of a repeated application of the following two steps. The first step is refers to a "greedy" assignment of nodes to communities, while favoring local optimizations in modularity. The second step focuses on definition of a new coarse-grained network with reference to the communities found in the first step. The above two steps are repeated till no further modularity-increasing reassignments of communities are possible. The advantage of Louvain method is that it enables the study of large networks owing to the fact that it typically takes less time to achieve modularities comparable to pre-existing algorithms. Also, there exist certain pitfalls to interpreting the community structure by the Louvain Method. In fact, these difficulties exist in all the modularity optimization algorithms.

References

[1] H.A. Simon. "The Architecture of Complexity". In: *Proceedings of the American Philosophical Society.* 106.6 (1962), p. 467.

[2] Réka Albert and Albert-László Barabási. "Statistical mechanics of complex networks". In: *Rev. Mod. Phys.* 74 (1 Jan. 2002), pp. 47–97. URL: https://link.aps.org/doi/10.1103/RevModPhys.74.47.

[3] M. Girvan M. E. J. Newman. "Statistical mechanics of complex networks-Finding and evaluating community structure in networks". In: *Phys. Rev. E 69, 026113 (2004)* 74 (1 Aug. 2003), pp. 47–97. DOI: 10.1103/PhysRevE.69.026113. URL: arXiv:cond-mat/0308217v1%20[cond-mat.stat-mech].

[4] Claudio Castellano Santo Fortunato. "Community Structure in Graphs". In: *Phys. Rev. E 69, 026113 (2004)* (Dec. 2007). DOI: 10.1103/PhysRevE.69.026113. URL: arXiv:0712.2716v1%20[physics.soc-ph].

[5] M. Girvan and M. E. J. Newman. "Community structure in social and biological networks". In: *Proceedings of the National Academy of Sciences* 99.12 (2002), pp. 7821–7826. ISSN: 0027-8424. DOI: 10.1073/pnas.122653799. eprint: https://www.pnas.org/content/99/12/7821.full.pdf. URL: https://www.pnas.org/content/99/12/7821.

[6] M. E. J. Newman and M. Girvan. “Finding and evaluating community structure in networks”. In: *Phys. Rev. E* 69 (2 Feb. 2004), p. 026113. DOI: 10.1103/PhysRevE.69.026113. URL: https://link.aps.org/doi/10.1103/PhysRevE.69.026113.

[7] “Defining and identifying communities in networks”. In: ().

[8] Pascal Pons and Matthieu Latapy. “Computing Communities in Large Networks Using Random Walks”. In: *Journal of Graph Algorithms and Applications* 10.2 (2006), pp. 191–218. DOI: 10.7155/jgaa.00124.

[9] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. “Finding community structure in very large networks”. In: *Phys. Rev. E* 70 (6 Dec. 2004), p. 066111. DOI: 10.1103/PhysRevE.70.066111. URL: https://link.aps.org/doi/10.1103/PhysRevE.70.066111.

[10] F. Wu and B. A. Huberman. “Finding communities in linear time: a physics approach”. In: *The European Physical Journal B* 38.2 (Mar. 2004), pp. 331–338. ISSN: 1434-6036. DOI: 10.1140/epjb/e2004-00125-x. URL: https://doi.org/10.1140/epjb/e2004-00125-x.

[11] M. E. J. Newman. “Finding community structure in networks using the eigenvectors of matrices”. In: *Phys. Rev. E* 74 (3 Sept. 2006), p. 036104. DOI: 10.1103/PhysRevE.74.036104. URL: https://link.aps.org/doi/10.1103/PhysRevE.74.036104.

[12] M. E. J. Newman. “Modularity and community structure in networks”. In: *Proceedings of the National Academy of Sciences* 103.23 (2006), pp. 8577–8582. ISSN: 0027-8424. DOI: 10.1073/pnas.0601602103. eprint: https://www.pnas.org/content/103/23/8577.full.pdf. URL: https://www.pnas.org/content/103/23/8577.

[13] Duncan J. Watts. “Networks, Dynamics, and the Small-World Phenomenon”. In: *American Journal of Sociology* 105.2 (Sept. 1999), pp. 493–527. URL: https://pdfs.semanticscholar.org/0c6f/bf40cf4%20fd1c8b4694e1e38bd03f8297e9cde.pdf.

[14] Vincent D Blondel et al. “Fast unfolding of communities in large networks”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 2008), P10008. DOI: 10.1088/1742-5468/2008/10/p10008. URL: https://doi.org/10.1088%5C%2F1742-5468%5C%2F2008%5C%2F10%5C%2Fp10008.

[15] Thomas Aynaud. *Louvain Community Detection*. URL: https://github.com/taynaud/python-louvain. (accessed: 03.22.2019).

[16] Meunier, D., Lambiotte, R., Fornito, A., Ersche, K. D., Bullmore. “Hierarchical modularity in human brain functional networks”. In: 11.3 (Oct 2009), p. 37. DOI: {10.3389/neuro.11.037.2009},journal={Frontiersinneuroinformatics}. URL: %7Bhttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC2784301/%7D.

[17] Aaron Clauset Benjamin H. Good Yves-Alexandre de Montjoye. “The performance of modularity maximization in practical contexts”. In: *American Journal of Sociology* (Oct. 2009). DOI: 10.1103/PhysRevE.81.046106. URL: arXiv:0910.0165v2%20[physics.data-an].