Community detection algorithms for real and artificial networks

Project Report - Group 3

Fábio Cruz
Nr. 96957
fabio.rocha.cruz
@tecnico.ulisboa.pt

Luís Fernandes
Nr. 97018
luis.martins.fernandes
@tecnico.ulisboa.pt

ABSTRACT

In this paper, we aimed to analyse and evaluate different community detection algorithms and compare their accuracy when applied to real and artificial networks.

The tested algorithms were the Louvain, Leiden, Girvan-Newman and Infomap methods, and they were tested against two real networks and two benchmark networks to better test the algorithm's accuracy.

We got good accuracy results for the Louvain, Leiden and Infomap methods and verified that the most computational efficient method was by far the Leiden method. We then concluded that the Leiden method would be the better option if we were to detect communities in an unknown network.

1. INTRODUCTION

Complex networks are an important area of research. With the exponential growth of science and technology in the past few years, a lot of work and attention has been pointed at this area from the scientific community. In the context of network theory, complex networks are essentially graphs that are composed by connections between different entities, and can be characterized by different topological measures, such as degree distribution, clustering coefficient, community structure and others.[5]

Community structure is defined on how the nodes of a network can be grouped into clusters/groups in such a way that a set of nodes is densely connected internally. These sets of nodes are called communities and can be found in several different types of networks such as biological, technological, and social networks. If a pair of nodes belongs to the same community the probability of them being connected is higher than if they don't. Usually, nodes are exclusive to only one community, but in specific cases a network can have overlapping communities where a node can belong to more than one, and some networks may not even have community structure such as random graphs and the Barabási-Albert model. [6]

Finding communities within a network can be computationally expensive since initially we usually don't know how many communities exist and how big they are. There are several community detection algorithms that can be divided essentially in two types: Agglomerative and Divisive methods. The idea behind Agglomerative methods is to start with disconnected nodes and add edges one by one in order to group up nodes with a high similarity into the same community, while in the Divisive methods we already start with a complete network with connected nodes and cut low similarity edges that tend to connect different communities.

Methods based on modularity optimization are also one of the most important type of community detection methods. Modularity is a measure of the structure of networks that calculates the strength of the division of the network into different sets of nodes. Networks with high modularity have nodes more densely connected with other nodes from the same community but have a weak connection with other communities. These types of methods aim to maximize modularity at each iteration by, for example, changing links between different pairs of nodes and calculating which configuration of the network gives a higher modularity value. However, modularity is known to have some problems such as the resolution limit for large enough communities, which makes the algorithm sometimes incapable of detecting smaller communities. [2]

In this paper we aim to analyse and evaluate different community detection algorithms and compare their accuracy when applied to real and artificial networks. In the next two sections we present what algorithms we will test and briefly explain how they work, and we will introduce the networks where we will apply those methods.

2. COMMUNITY DETECTION ALGORITHMS

We chose to analyse and compare some popular community detection methods that are known to give good results and be efficient for large complex networks, but also that have different types of implementation and ideologies.

2.1 Girvan-Newman algorithm

The first algorithm we decided to use was the Girvan-Newman algorithm, named after the popular complex network researchers Michelle Girvan and Mark Newman. The main idea of this algorithm is to sequentially remove edges/links that connect nodes that hopefully belong to different communities (Divisive method).

Divisive methods use centrality measures that are high for nodes that belong to different communities and low for the ones that belong to the same one. Girvan-Newman uses link betweenness as its centrality measure. The higher the link betweenness of an edge, the higher the probability of that edge connecting different communities.

The steps to run this algorithm can be simplified to this:

1. Compute centrality/betweenness for each link
2. Remove the link with the highest centrality value. In case of a tie, choose randomly
3. Recalculate centrality values after removing the edge
4. Repeat step 2 and 3 until all edges are cut

At the end of the algorithm we get a dendrogram. To obtain the final result with the community structure of the network we need to "cut" the dendrogram at a certain level. One way to choose at what level we should cut is to calculate the modularity for a set of different levels and choose the best one. The computational complexity of this algorithm is approximately $O(m^2n)$ for the worst case, where n is the number of nodes, and m the number of edges of the network.[2][7]

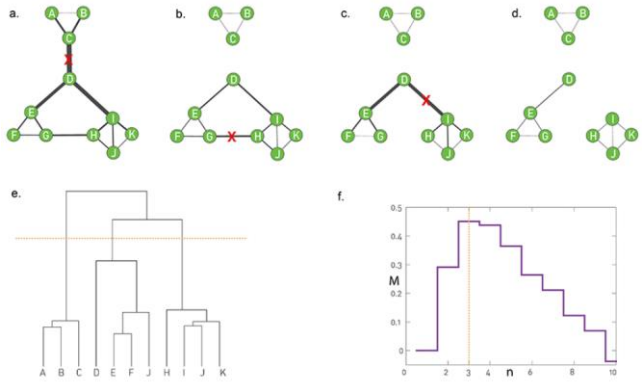


Figure 1 – Representation of the Girvan-Newman algorithm, taken from the Barabási book Network Science [2]

2.2 Louvain method

The next algorithm we decided to test was the Louvain method, created by Blondel et al. from the University of Louvain. This is an Agglomerative method that uses modularity optimization to find communities and can be used on very large networks since it is very efficient from a computational point of view (complexity of $O(m)$).

Initially, the algorithm assigns a different community for each node of the network. Then for each node it calculates the modularity value if the node is changed from the current community to a neighbour community. The change will occur if modularity is maximized that way, otherwise nothing changes. This process is repeated until no more improvements to the modularity of the network can be done. After this, the algorithm transforms the current communities into nodes and reapplies the previous procedure. Again, this is repeated until no more improvements are accomplished in the network. Because of this the Louvain method is capable of finding communities inside communities, and it is easy to implement and very time efficient. However, it has some problems…[2][8]

2.3 Leiden method

One of the Louvain method problems is that it sometimes can discover weekly connected communities. To solve this problem, T.A. Traag has created the Leiden method which is also faster than the previously mentioned method.

This algorithm is very similar to Louvain. In the first phase Leiden only visits the nodes whose neighbourhood has changed instead of all the nodes, and it adds a new phase after the ones explained before, where it essentially tries to refine the partitions made in the second phase where new communities can be created from the previous existing communities. This refinement means that a node may be merged with a randomly chosen community which increases some quality function. [3]

2.4 Infomap

Last but not least, we decided to add to our set of algorithms the Infomap method to compare it with the other more known and used algorithms such as the Girvan-Newman and the Louvain, and see if Infomap is also a viable option in terms of accuracy and time complexity.

This algorithm is a little complex to describe, but essentially the Infomap algorithm tries to minimize a cost function and is based on data compression for community detection. It is accomplished by encoding the best trajectory of a random walker in the network, using something called a map equation.

Every node is labelled with a code. If a random walker walked inside our network, we would want to describe its path with the least number of symbols possible using those codes. But we know that normally a random walker tends to stay "trapped" for longer when it is inside a community, so we can partition the network and give a code to every community as well so we can optimize and find the shortest message possible with those community codes. Essentially, we associate the paths that are more used by the random walker with edges between nodes that belong to the same community, and paths or edges that are not travelled a lot are probably connecting different clusters of nodes. If we partition the network into too many modules the message encoded becomes bigger than what it needs to be, so this way we can find an optimal partition that assigns nodes to modules in such a way that the information needed to compress the movement of our random walkers is minimized (the cost function).

The reason this method is also very good in terms of time complexity ($O(m)$) is because instead of brute forcing this algorithm, which would be bad, they use a variation of the Louvain method to help find good partitions.[2][9]

3. METHODS

To implement and apply all these community detection methods we used NetworkX and CDlib[12], a community detection library for python that allows to extract, compare and evaluate communities from complex networks. This library already has implemented most of the algorithms for community detection as well as lots of evaluation measures for the networks and partitions, so we didn't have to implement the algorithms from scratch.

Regarding the networks that were used in this project, we used two real networks from the Stanford Large Network Dataset Collection**Erro! A origem da referência não foi encontrada.**. The first one was a small/medium sized network with around a thousand nodes that used email data from a large European institution[14]. The connections between nodes in this network represent communication between institution members. This is an undirected network with a ground-truth community structure so we can easily compare the accuracy of the tested community detection methods that were applied to the network. The second network is much bigger, with around 317 000 nodes and a million edges and is also an undirected network.[15] It is a computer research bibliography that contains a big list of research papers in computer science. Two authors/nodes are connected if they publish a paper together. This dataset also has ground-truth communities, but we decided to ignore them and use only the large network to test if the better algorithm would do a good partition, by testing the modularity of the resultant communities of this network.

The idea was to first test the different algorithms in the smaller network and the artificial benchmark networks, and then only use the most accurate and fastest algorithms in the larger network and analyse the model modularity with this network labels.

4. RESULTS

Finally, regarding the results of our work we first started by analysing some properties of the networks that we used. Then since the Girvan-Newman has a parameter to determine at what level the "cut" is done to choose the community structure we tested what the best parameter was for different values of that parameter for a certain network. The same was done for Infomap since it also has a parameter for the number of trials that we can do in this algorithm. After that we could finally run the different community detection algorithms for the networks and compare the results, as well as analysing some properties of the final community structures.

We also compared the running times of the four algorithms to check if they followed their theoretical computational complexity.

4.1 Networks

Network	Nodes	Edges	Expected Communities	Average Degree	Average Path Length	Clustering Coefficient
Email	1005	16706	42	33	2,587	0,399
LFR	500	1076	16	4	5,6865	0,1891
SBM	450	22054	4	98	1,7948	0,3022
DBLP	317080	1049866	13477	7	-	0,632

Figure 2 – Real and artificial network analysis results

In the table shown above we can see some important characteristics of the networks used in this project. Besides the number of nodes, edges and the number of expected communities for the real networks, we also computed a few other properties. The Average Degree (AD) shows us on average how many connections the nodes of the network have. The Average Path Length (APL) represents the average shortest distance from a pair of nodes, which basically means the least amount of jumps we must do to reach one node from another on average. Finally, we also computed the clustering coefficient of the network, which is related to how close nodes from the same cluster/community are from each other and further away from nodes of different communities.

First, as we said before, the Email network is not very large, having only 1005 nodes and 16706 edges, and has as ground-truth 42 communities. If we check the connections of nodes that this network has we can clearly see that this network is not fully connected. It has a big cluster of nodes in the center of the network and some isolated islands of nodes with only connections to themselves and no connections to other nodes (emails sent from one person to themselves). This means the Average Path Length shown in the table above represents only the main cluster of nodes of the network, since the other disconnected nodes will automatically have an APL of 0. We decided to ignore these nodes in this metric. In the image below we can also verify that this network's degree distribution follows a power-law, meaning it can be considered a scale-free network.

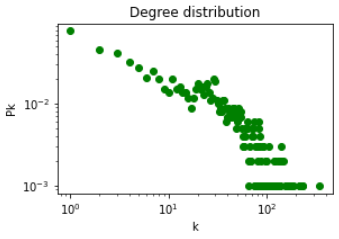


Figure 3 – Power-law degree distribution for the Email network

As we said before in this paper, we also decided to apply our algorithms to some artificial benchmark networks, so we can confirm again the accuracy of the methods with more networks that have ground-truth communities and are well known in the area of complex networks. The benchmark networks we decided to use were the LFR (Lancichinetti–Fortunato–Radicchi) benchmark and the SBM (Stochastic block model) benchmark.

The LFR benchmark essentially generates nodes that follow a power-law distribution, it assigns each node to a community where the community sizes are fixed and also

follow a power-law, and finally takes into consideration the internal (1-μ)ki and external degrees (μki) of the nodes to randomly attach free nodes according to those degrees. This μ is a free parameter that can be tested to see what is the value that gives the best results, as we will see later in this paper.[9]

The SBM benchmark is a generative model for random graphs where we initially decide what the number of communities and their sizes are, and the probability matrix that defines the probability of generating a connection between different nodes that can belong to different communities. We can also change the probabilities of those connections to test what are the best parameters for the results of our algorithm partitions.

Last but not least, our last real network has 371080 nodes and around a million edges! For this reason, we will only apply the fastest algorithms that we verified that give decent results in the previous networks. One interesting thing that we can see in the table above is that the clusters of this network are actually well defined since the clustering coefficient is the highest of all the networks tested. Unfortunately, we couldn't run the Average Path Length for this large network since this computation is heavy, the computational complexity of this metric can escalate to $O(n^2)$ in the worst cases. We can also verify again that the degree distribution of this real network follows a power-law.

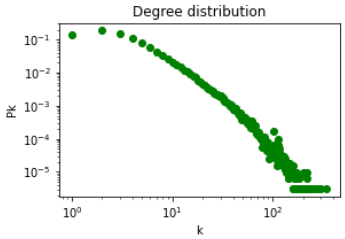


Figure 4 – Power-law degree distribution for the DBLP network

4.2 Best Girvan-Newman parameter

Like we mentioned, before applying the Girvan-Newman algorithm to the networks we need to see what the best parameter value for the level is, where we will cut our dendogram of this method. Theoretically, the higher the level where we cut the higher the number of communities that the algorithm retrieves. As we can see in the graph below that shows the Girvan-Newman modularity for the Email network the values are extremely low for this network (the modularity is always below 0,01). We think that this happens because of the disconnected nodes that exist in this network. These nodes are probably present in the higher levels of the dendogram, so when we do the initial cuts the modularity between these disconnected nodes is very low since the modularity is calculated by how densely connected nodes from the same community are, but in this case we have isolated communities with only one node each.

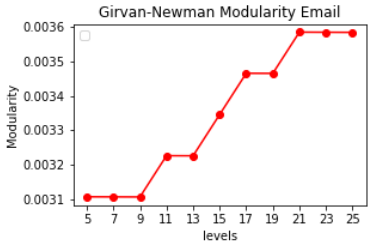


Figure 5 – Girvan-Newman modularity values for different level parameters, applied to the Email network

In the benchmark networks this didn't happen. For the LFR benchmark we had the best modularity value for the Girvan-Newman algorithm around 0,86 when cutting the dendogram in level 15, which is obviously much better than the results we got for the Email network. For the SBM benchmark the best modularity value was around 0,46 for level 3. We think that the SBM modularity values are lower than the LFR ones because SBM only has 4 ground-truth communities while LFR has 16. Another possible reason is that while the generation of the LFR network is based from scratch on nodes and communities that follow a power-law which makes the ground-truth communities already well densely connected regarding nodes from the same community and sparsely connected with nodes from different communities, while the SBM is a statistical model that uses probabilities for random connections between nodes, so there is a higher probability that these communities are not so well connected as the LFR ones.

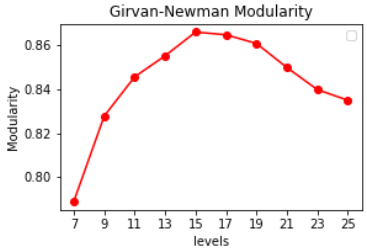


Figure 6 - Girvan-Newman modularity values for different level parameters, applied to the LFR benchmark network

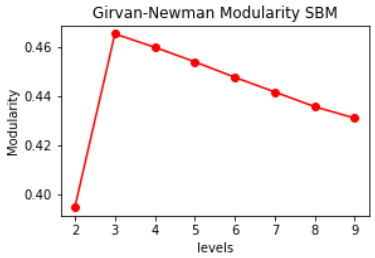


Figure 7 - Girvan-Newman modularity values for different level parameters, applied to the SBM benchmark network

4.3 Best Infomap parameter

We also tested the Infomap parameter for the number of trials to run this algorithm. As we can see from the graph below generated

in our experiment, the ideal modularity has a value of around 0.42 for 4 trials, so this is the parameter value that we will use when applying the algorithm to the real networks.

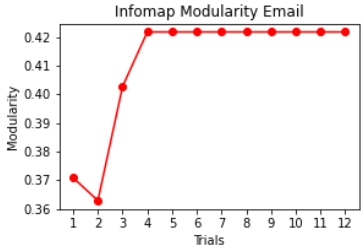
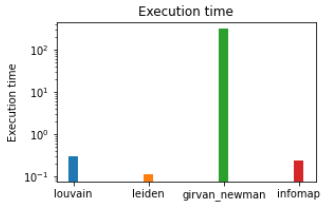


Figure 8 - Infomap modularity values for different number of trials parameters, applied to the Email network

4.4 Running times

After running the Louvain, Leiden, Girvan-Newman and Infomap methods with the optimal parameters (when applied) we wanted to verify if the theoretical time complexities of these algorithms would be reflected in our experiments. As we can see from the images below, the Girvan-Newman algorithm was by far the least efficient of the ones tested, as it was expected since its time complexity for the worst-case scenarios is $O(m^2n)$, where m is the number of edges of the network and n the number of nodes. This makes sense since the running time of the algorithm grows faster when the number of edges is high, and our Email network has a considerable number of edges (around 16 thousand), even though the number of nodes is relatively small. The Louvain and Infomap methods both have a similar running time, which also makes sense since their computational complexity is the same ($O(m)$). Finally, the most efficient algorithm for this Email network was the Leiden method. Again, this can be theoretically confirmed since this method is an optimization of the Louvain method and is supposed to be more efficient, as well as giving similar or better results for the partitions.



Times (s)	Louvain	Leiden	Girvan-Newman	Infomap
Email	0,331	0,070	382,820	0,261

Figure 9 - Execution times of the four algorithms

4.5 Community internal properties

Now we will check some internal evaluation properties of the community structures given by the different algorithms when applied to the Email network. The properties that we checked for the communities are represented in the table below. The Average Community Size and Average Path Length for all the communities are self-explanatory, they represent the averages of the sizes of all communities returned by the algorithms and the average of all shortest path lengths. The Average Internal Degree is the average number of connections inside the communities, the Average Embeddedness is the ratio between the degree of a node to other nodes inside the community against the degrees to nodes outside and finally the Average Transitivity is the average clustering coefficient of a community nodes in relation to their connection within the community itself.

The most interesting thing we can verify in these properties is that the Girvan-Newman method has much lower values for APL, AID and AT when in comparison with other algorithms. We think the reason for this to happen is, again, the existence of disconnected nodes. In the Girvan-Newman all of these nodes will represent a community with only one node, meaning these communities values of Average Path Length, Average Internal Degree and Average Transitivity will be 0 or close to it, consequently lowering significantly these averages for all the communities of the network. The Average Embeddedness is not affected since these isolated nodes have only connections to themselves, meaning the value for the Average Embeddedness of these isolated communities will be 1 instead of 0.

In general, the Average Embeddedness is very high for all algorithms. This means that most nodes have a much higher number of connections to other nodes inside the same community in comparison to connections to nodes outside the cluster, meaning the algorithms did a decent job partitioning the networks in regard to this particular measure.

Algorithm	Communities	ACS	APL	AID	AE	AT
Louvain	28	36	0,669	7,141	0,903	0,177
Leiden	27	37	0,641	7,175	0,917	0,159
Girvan-Newman	23	44	0,193	3,236	0,905	0,018
Infomap	34	30	0,691	6,322	0,855	0,173

ACS - Average Community Size
APL - Communities Average Path Length
AID - Average Internal Degree
AE - Average Embeddedness
AT - Average Transitivity

Figure 10 - Community structure internal properties for the four algorithms

4.6 Accuracy

In terms of accuracy, we decided to use the Normalized Mutual Information (NMI), where we compared each partition with the ground-truth labels that we had available for the Email network and the benchmark networks. This measure will then compare the correlation/similarity between the communities returned by the best versions of our algorithms and the communities that were expected to be returned.

We can conclude for the Email network that the best method in terms of accuracy was Infomap, with a NMI of 62%. Louvain and Leiden were also close candidates with 54% and 58%, respectively. The least accurate algorithm by far was the Girvan-Newman giving a correlation of only 7,5% with the ground-truth labels for this network. We think the reasons for this to happen are same that were already explained before in this paper.

In general, the accuracy of the algorithms was not incredible, but we think this is justified by the fact that the structure of this network is odd since it has several isolated communities of single nodes that can be hard do identify by some of these algorithms, and also the clustering coefficient was initially not very high as well, meaning the clusters in this network were not very well defined.

Accuracy	Louvain	Leiden	Girvan-Newman	Infomap
Email	0,537	0,579	0,075	0,6161

Figure 11 – NMI accuracy values for all the algorithms, applied to the Email network

For the LFR benchmark network we decided to test the accuracy of the algorithms against different μ parameters for this benchmark, in order to see what gives the best results.

We can see that all the algorithms start to decline in their accuracy when μ is 0,2 or higher. This makes sense because the higher this parameter is, the higher the probability of generating links between nodes of different communities and less links between nodes of the same community, which will decline the modularity of the network and consequently the accuracy of the algorithms when partitioning this network. Of all the four algorithms, the one whose accuracy decline the slowest was the Infomap method.

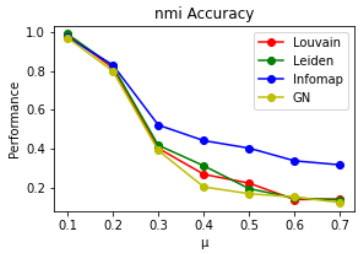


Figure 12 - NMI accuracy values for different LFR parameters, comparing the performance of all four algorithms

For the SBM benchmark network we also changed around some parameters: we created a network with four communities but tested it for different probabilities of links of different communities connecting. This means, again, that the higher the value of μ, the more likely that the communities are badly clustered. In this benchmark we got the best accuracy results with the Louvain and Leiden methods.

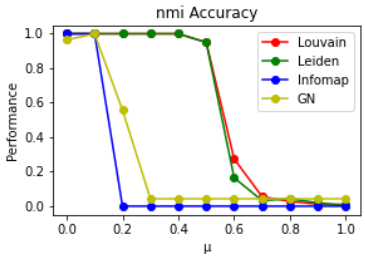


Figure 13 - NMI accuracy values for different SBM parameters, comparing the performance of all four algorithms

4.7 Modularity

Now regarding the modularity of our algorithms for each network (real and artificial), we can see that the algorithms that got the best modularity results for the Email and benchmark networks were the Louvain, Leiden and Infomap methods, all having very similar modularity results. This makes sense because both Leiden and Infomap methods are somewhat based on the Louvain method or use it, meaning the results for the partitions of these algorithms with be similar.

For the large DBLP network we decided to not use the Girvan-Newman due to its computational inefficiency and poor accuracy results for the Email network, and for the other algorithms we got very decent results with modularity values around 0,8 for all of them, meaning that the Louvain, Leiden and Infomap are good for large networks with a very high number of nodes and edges. They are also very time efficient since we verified that the Louvain and Infomap methods took around 3 minutes to partition this large dataset, and the Leiden method was extremely fast taking only around 12 seconds.

Modularity	Louvain	Leiden	Girvan-Newman	Infomap
Email	0,431	0,433	0,004	0,422
LFR	0,864	0,869	0,866	0,862
SBM	0,141	0,140	~0	~0
DBLP	0,820	0,832	-	0,819

Figure 14 – Modularity values for all four algorithms, applied to all networks

Another way to prove that the Louvain, Leiden and Infomap are similar is by checking their correlation using the Normalized Mutual Information. After checking this for our methods, we verified that the NMI value for the pairs (Louvain, Leiden), (Louvain, Infomap) and (Leiden, Infomap) are all high, around the 80% mark when relating Louvain and Leiden and the 70% mark when relating Infomap with one of the other two. When comparing the correlation of Infomap with any other algorithm the result for NMI is pretty low (around 20% only).

5. CONCLUSIONS

After all our experiments and results achieved, we can take some conclusions regarding these community finding algorithms.

The two main things we concluded in this paper to find out what is the best community detection algorithm, if there is one, were regarding execution times and accuracy values.

For the first of the two, it is clear from the results that the most efficient algorithm of the ones that were tested was the Leiden method, since it only took about 12 seconds to run on a network with over a million links. The Louvain and Infomap methods took around 3 minutes which is also not too bad for this network, but is not even close to the computational efficiency of the Leiden method.

For the accuracy of the algorithms when applied to these networks, we can conclude that Louvain, Leiden and Infomap are all similar in terms of accuracy results. The only

exception was the Infomap accuracy for the SBM benchmark network, where it dropped relatively fast when compared with the other two.

In short, if we had to pick one of the algorithms to detect communities on an unknown network without any previous knowledge of ground-truth communities or the network's structure, we would choose the Leiden method since it gives similar accuracy results to Louvain and Infomap, yet is much faster.

For future work, it would be interesting to test different types of algorithms, such as methods based on overlapping communities where nodes can belong to more than one community. Unfortunately, we were not able to implement this for this paper since these algorithms are extremely inefficient for large networks, but it would be interesting to see if the accuracy results for a overlapping network would be higher than the algorithms tested in this paper.

It would also be useful to test the Email network in particular, with an algorithm that verifies nodes that don't have connections to other nodes before building up the communities, since this was one of the major problems for the Girvan-Newman method, in our opinion.

Another good algorithm to use would be one that considers large networks that may have nodes that are simply not connected to anything, not even to themselves.

Finally, we would like to end this paper by referencing an interesting alternative for the calculation of modularity that would be extremely interesting to test with more time and resources. Very recently (October 2021) came out a paper called "Robustness modularity in complex networks" from several popular researchers, including Fortunato, where they propose an alternative calculation for the modularity of network partitions that solves some of the problems with classic modularity such as the resolution problem. They essentially compute the robustness of a network partition by creating entropy on the network with a probability p. They check at what point the network "breaks" and use that to calculate a new modularity measure that they call Robustness Modularity (RM). It looks to be a very interesting paper and this new measure seems to give out very good results for several real networks that were tested, as well as in benchmark networks such as the SBM that was used in this paper too.

6. ACKNOWLEDGMENTS

Our thanks to all the professors of the course for the help and material provided, and to all the authors of the papers/articles present in our references.

7. REFERENCES

[1] Course materials regarding community detection in complex networks

[2] Barabási, A. L. (2013). Network science. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 371(1987), 20120375. https://doi.org/10.1098/rsta.2012.0375

[3] Jayawickrama, T. D. (2021, February 1). *Community Detection Algorithms - Towards Data Science*. Medium. https://towardsdatascience.com/community-detection-algorithms-9bd8951e7dae

[4] Silva, F. N. (2021, October 5). *Robustness modularity in complex networks*. ArXiv.Org. https://arxiv.org/abs/2110.02297

[5] Wikipedia contributors. (2021, August 25). *Complex network*. Wikipedia. https://en.wikipedia.org/wiki/Complex_network

[6] Wikipedia contributors. (2021b, September 10). *Community structure*. Wikipedia. https://en.wikipedia.org/wiki/Community_structure

[7] Wikipedia contributors. (2021a, July 22). *Girvan–Newman algorithm*. Wikipedia. https://en.wikipedia.org/wiki/Girvan%E2%80%93Newman_algorithm

[8] Wikipedia contributors. (2021c, September 8). *Louvain method*. Wikipedia. https://en.wikipedia.org/wiki/Louvain_method

[9] Wikipedia contributors. (2020, July 20). *Lancichinetti–Fortunato–Radicchi benchmark*. Wikipedia. https://en.wikipedia.org/wiki/Lancichinetti%E2%80%93Fortunato%E2%80%93Radicchi_benchmark

[10] Wikipedia contributors. (2021e, October 30). *Stochastic block model*. Wikipedia. https://en.wikipedia.org/wiki/Stochastic_block_model

[11] Rita, L. (2020, April 12). *Infomap Algorithm - Towards Data Science*. Medium. https://towardsdatascience.com/infomap-algorithm-9b68b7e8b86

[12] https://cdlib.readthedocs.io/en/latest/

[13] https://snap.stanford.edu/data/#communities

[14] https://snap.stanford.edu/data/email-Eu-core.html

[15] https://snap.stanford.edu/data/com-DBLP.html