Finding community structure

Martí de las Heras Sonia Petrini

November 24, 2021

1 Introduction

The objective of this project is to explore a variety of community detection algorithms, and see some of the strengths and weaknesses for all of them. In order to do so, we compute 4 metrics over the obtained communities on 4 different graphs. Two of them are provided by the `igraphdata` package, while the other two where artificially created by merging networks in which members are tightly connected, and communities with less within-group connections.

- *Macaque Network*: Visuotactile brain areas and connections of the macaque monkey.
- *Hospital Network*: Records of contacts among patients and various types of health care workers in the geriatric unit of a hospital in Lyon, France, in 2010.
- *Artificial Network from ER Graphs.*
- *Artificial Network from Complete Graphs.*

Finally, to conclude, we have applied those concept on a bigger network `Wikipedia.gml`.

2 Results

In the following, we present the tables with the 4 considered scores for each of the 9

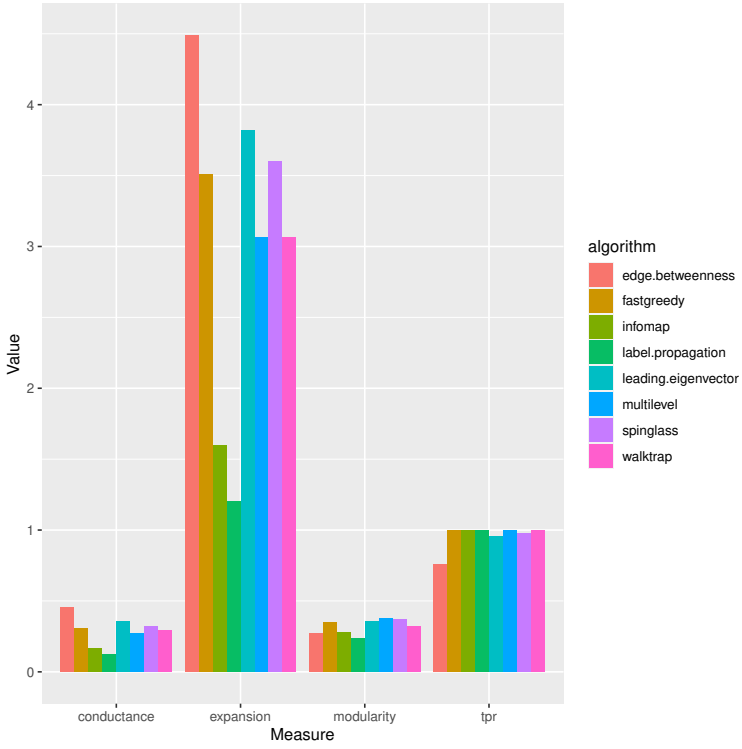

Figure 1: **Macaque Network**: barplot of resulting measures for each algorithm.

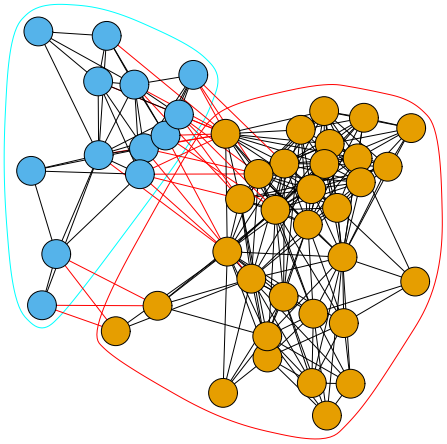

Figure 2: **Macaque Network**: best clustering configuration, according to *label propagation*.

algorithm	clusters	time	tpr	expansion	conductance	modularity
edge.betweenness	13	0.04	0.76	4.49	0.46	0.27
fastgreedy	3	0.00	1.00	3.51	0.31	0.35
label.propagation	2	0.00	0.98	1.20	0.12	0.22
leading.eigenvector	4	0.01	0.96	3.82	0.36	0.36
multilevel	3	0.00	1.00	3.07	0.27	0.38
spinglass	4	0.25	1.00	3.29	0.30	0.37
walktrap	4	0.00	1.00	3.07	0.30	0.32
infomap	3	0.00	1.00	1.60	0.17	0.28

Table 1: Resulting measures for **Macaque Network**: Visuotactile brain areas and connections.

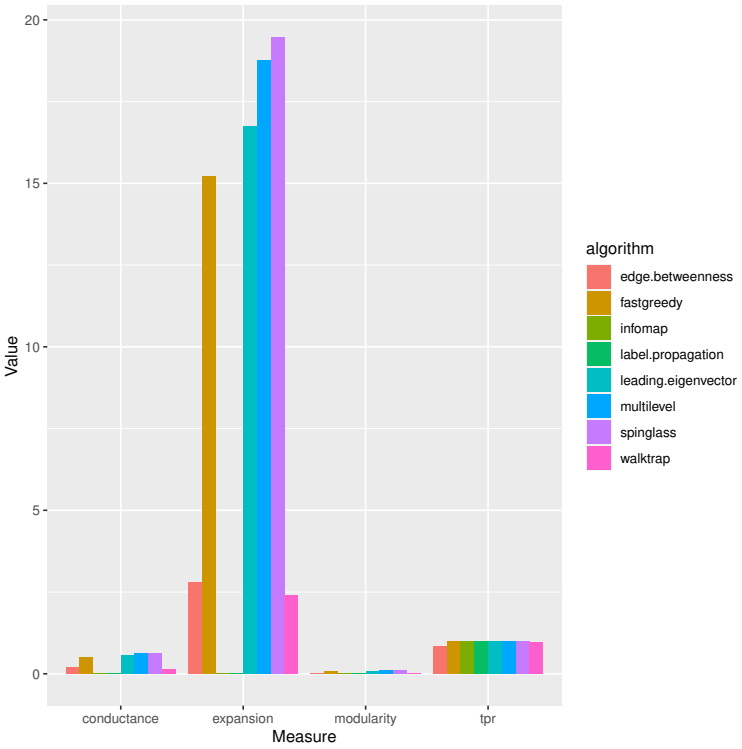

Figure 3: **Hospital Network**: barplot of resulting measures for each algorithm.

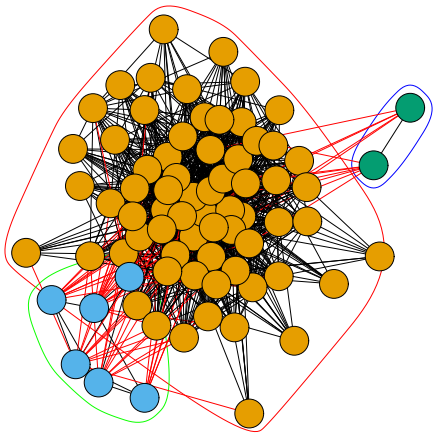

Figure 4: **Hospital Network**: best clustering configuration, according to *walktrap*.

algorithm	clusters	time	tpr	expansion	conductance	modularity
edge.betweenness	12	0.94	0.85	2.80	0.18	0.00
fastgreedy	3	0.00	1.00	15.23	0.50	0.09
label.propagation	1	0.00	1.00	0.00	0.00	0.00
leading.eigenvector	3	0.01	1.00	16.75	0.55	0.08
multilevel	4	0.00	1.00	18.77	0.62	0.10
spinglass	4	1.25	1.00	19.39	0.64	0.10
walktrap	3	0.00	0.96	2.40	0.12	0.02
infomap	1	0.0	1.00	0.00	0.00	0.00

Table 2: Resulting measures for **Hospital Network**: Records of contacts among patients and various types of health care workers.

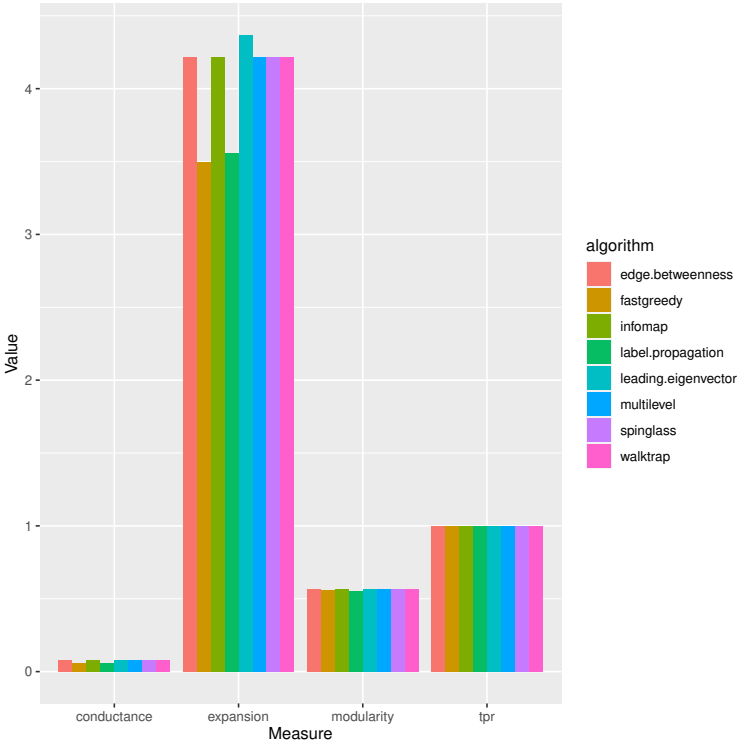

Figure 5: **Complete Network**: barplot of resulting measures for each algorithm.

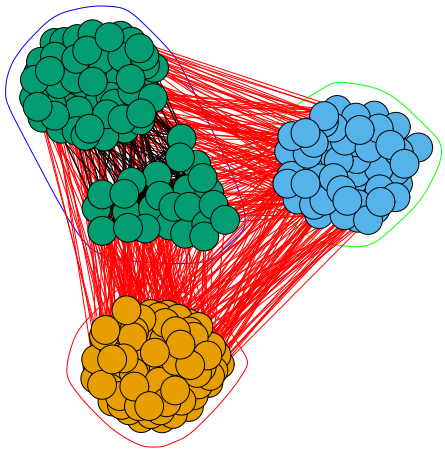

Figure 6: **Complete Network**: best clustering configuration, according to *fastgreedy*.

algorithm	clusters	time	tpr	expansion	conductance	modularity
edge.betweenness	4	36.91	1.00	4.21	0.08	0.57
fastgreedy	3	0.01	1.00	3.50	0.06	0.56
label.propagation	3	0.00	1.00	3.56	0.06	0.55
leading.eigenvector	4	0.01	1.00	4.37	0.08	0.56
multilevel	4	0.00	1.00	4.21	0.08	0.57
spinglass	4	2.95	1.00	4.21	0.08	0.57
walktrap	4	0.02	1.00	4.21	0.08	0.57
infomap	4	0.04	1.00	4.21	0.08	0.57

Table 3: Resulting measures for the **Complete Network**, with 4 original communities and difficulty = 30, meaning only one edge between each cluster.

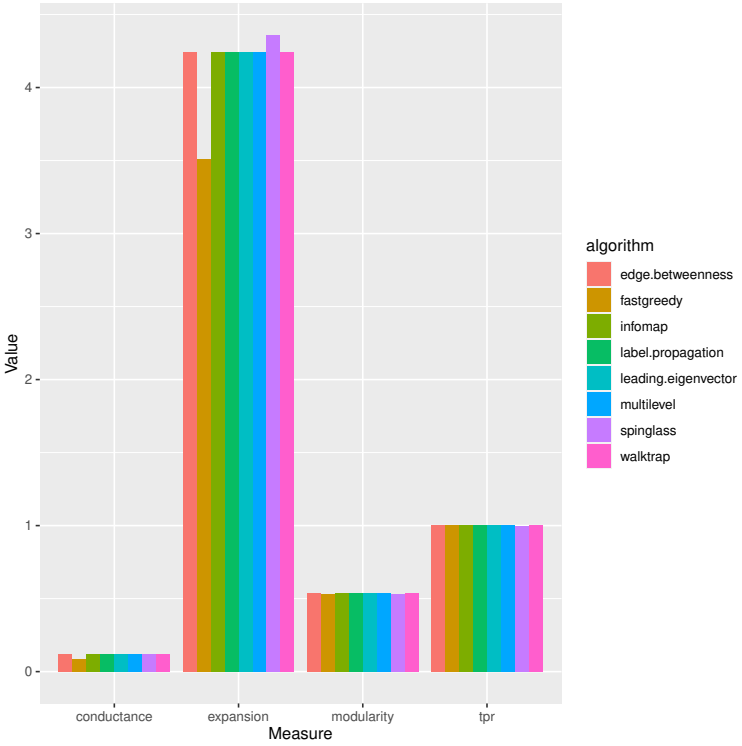

Figure 7: **ER Network**: barplot of resulting measures for each algorithm.

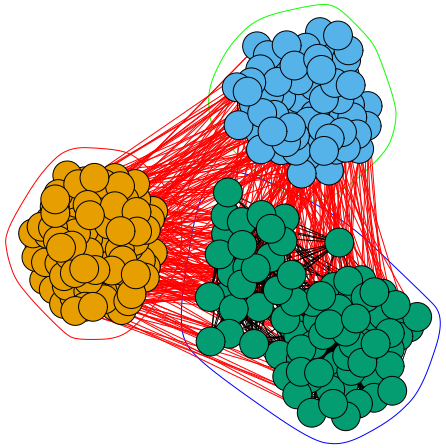

Figure 8: **ER Network**: best clustering configuration, according to *fast greedy*.

algorithm	clusters	time	tpr	expansion	conductance	modularity
edge.betweenness	4	32.28	1.00	4.24	0.12	0.54
fastgreedy	3	0.01	1.00	3.51	0.09	0.53
label.propagation	4	0.00	1.00	4.24	0.12	0.54
leading.eigenvector	4	0.02	1.00	4.24	0.12	0.54
multilevel	4	0.00	1.00	4.24	0.12	0.54
spinglass	5	1.90	1.00	4.36	0.12	0.53
walktrap	4	0.01	1.00	4.24	0.12	0.54
infomap	4	0.03	1.00	4.24	0.12	0.54

Table 4: Results obtained for the **ER Network**, with 4 original communities and difficulty = 30, meaning 30 additional edges.

2.1 Task 2: Wikipedia

In this task we have analyzed a Wikipedia graph provided in the lab. Vertices of this network are wikipedia pages and the label of each is the title of the Wikipedia page.

First of all, to have the main insights of this network, we computed some general statistics as we can see in figure 9. It is a huge graph composed by 27475 pages, linked by 85729 edges. Another interesting graph property is to check the degree distribution of the network. We can clearly observe, as we should expect, that the degree of the Wikipedia follows a *Zipf's law* distribution. Which helps us to understand the context and behaviour of this concrete network and have more insights on how the pages are connected.

We have applied 4 different community detection algorithms over our network. The results obtained can be seen in the table 5. Those metrics are acquired after pruning singletones obtained in the clustering procedure. Therefore, those values differ from the first clustering iteration over the whole graph. As we can observe, the `fast greedy algorithm` is the one with best values. Less clusters (with higher number of vertex in each of them) and also, a lower expansion and a higher modularity, showing better intra-cluster quality than other algorithms.

Finally, to check if the communities obtained have any sense, we have plotted word clouds with the most frequent words from the wikipedia pages within a community. As we can see in figure 10, they do have sense. For example, in the first wordcloud, we could interpret that this cluster is refering to Chemical wikipedia pages, second one to telecommunications, forth one into mathematical theory, etc. Although this is only a sample of 12 groups, we can assume good clustering results in the wikipedia communities.

Algorithm	Clusters	Expansion	Modularity
fast greedy	223	0.4736410	0.7397027
walktrap	1316	1.0871562	0.6660896
label propagation	812	0.8170285	0.6903954
infomap	1408	1.9825671	0.6452076

Table 5: Main statistics of the 4 algorithms applied in Wikipedia dataset.

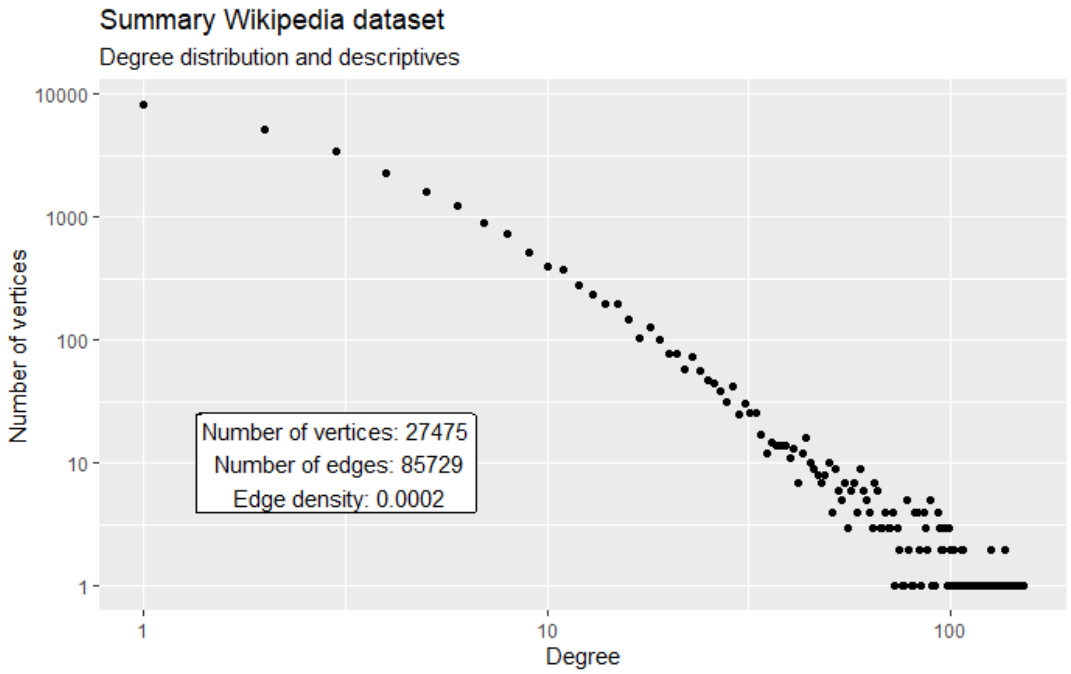

Figure 9: Degree distribution and main statistics from the whole `Wikipedia.gml` graph.

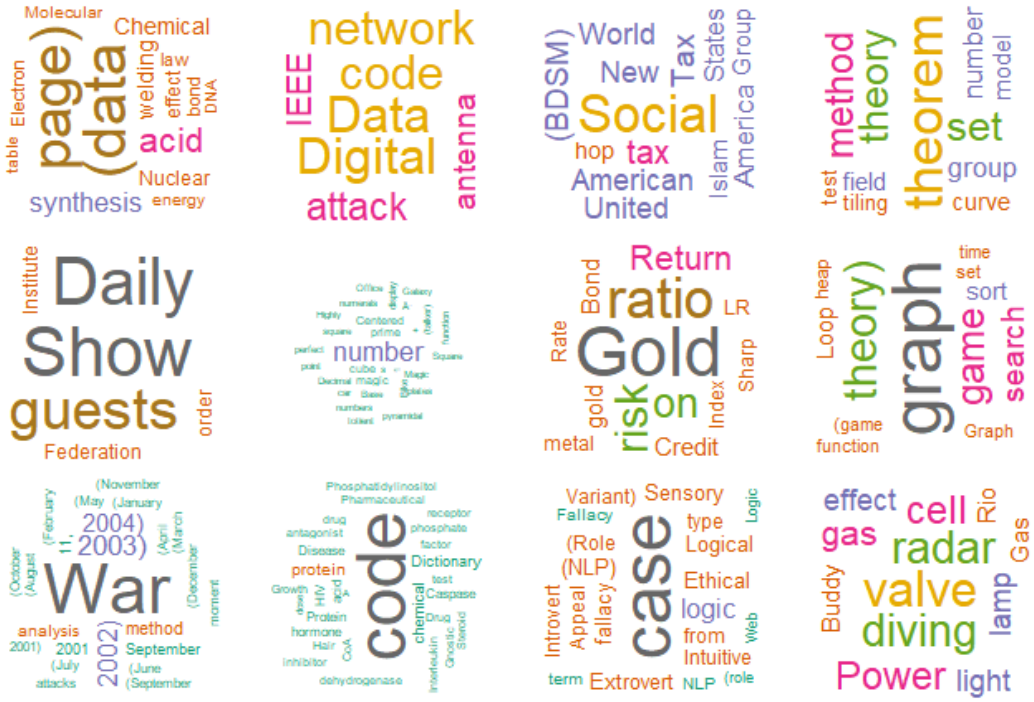

Figure 10: Visualization of the words with higher frequency on 12 clusters computed by `fast greedy` community detection algorithm.

3 Methods

3.1 Metrics Definition

While the `modularity()` function is already implemented in `igraph`, we computed *triangle participation ratio* (tpr), *expansion*, and *conductance* for each metrics and each network according to their definitions:

- `Conductance`: for each edge in the graph, we check whether the head and the tail nodes share the same membership. Thus, we add the edge to the count of within-cluster edges of the relative group if they do, and to the between-cluster edge count otherwise. Then, since
- `Expansion`: to compute the expansion of the whole graph, we make use of the `igraph` function `crossing`, which counts whether an edge inside the partition is pointing out the cluster.To each of these edges correspond 2 stubs, one for each node forming the edge. Thus, after simplifying the numerosity of the clusters, we can compute expansion as a weighted average of the expansions of the single clusters as follows: $e = \frac{2\sum f_c}{n}$, where n = number of vertices and f_c edges pointing out the cluster c.
- `Triangle Participation Ratio`: for this metric we also leveraged a function already defined in `igraph`, namely (`triangles`). We retrieve the triangles in a cluster c and count the number of unique nodes that compose those triads. Lastly, we compute the mean adding the number of nodes in triangles and dividing for the total number of nodes.

3.2 Artificial Networks

Community detection is typically an unsupervised learning process. Thus, to evaluate the performance of the algorithms against a known truth, we implemented two functions to generate artificial networks with communities fixed a priori. First, `merge_communities_full()` assumes a **fully connected** community design, where groups are complete graphs. Second, `merge_communities_ER()` allows one to create a network of **Erdos-Renyi** graphs, tuning through *ERp*, the probability of connection between two nodes, the sparsity of the linkages within the communities. We will refer to the Networks we build with these functions as *"Complete"* and *"ER"* Networks. For the purposes of the analysis, both these Networks

are created with 4 groups having populations: (94,53,25,62), in order to compare the algorithms' performances on the same fixed Network size. To create the communities in the ER Network, we set the probability of appearance of a new connection *ERp* to 0.6.

3.2.1 Clustering Difficulty

Both functions allow to tune the ***difficulty*** of the clustering task, choosing the number of connections between the original groups. The value of the *difficulty* parameter is multiplied by 5 to get the total number of between-cluster edges to add. For both the "Complete" and the "ER" Network, difficulty is set to 100, meaning that 500 new edges were created between the groups. Then, to get an understanding of the ***performance*** of the algorithms for increasing connection between the original clusters, we plot the number of estimated clusters for a grid of *difficulty* values between 10 and 100, with a step of 5. The results for 4 groups with population sizes (50,47,35,28) are shown in 11 and 12 for the completely connected communities, and in 13 and 14 for those based on ER graphs, with *ERp*=0.4.

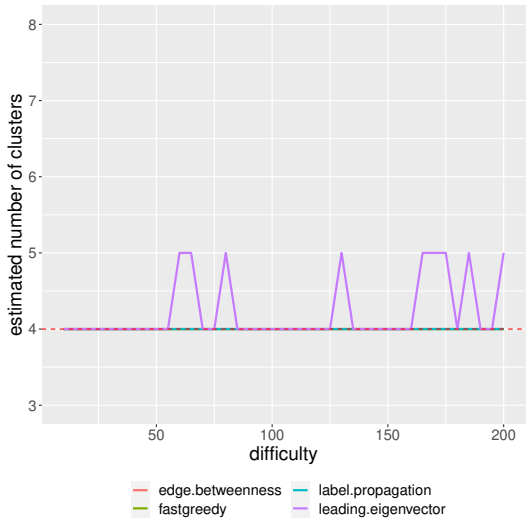

Figure 11: Estimated number of clusters for the **"Complete" Network**, for increasing difficulty.
Real number of groups: 4.
Metrics: *edge betweenness*, *fastgreedy*, *label propagation*, *leading eigenvector*

Figure 12: Estimated number of clusters for the **"Complete" Network**, for increasing difficulty.
Real number of groups: 4.
Metrics: *multilevel*, *spinglass*, *walktrap*, *infomap*

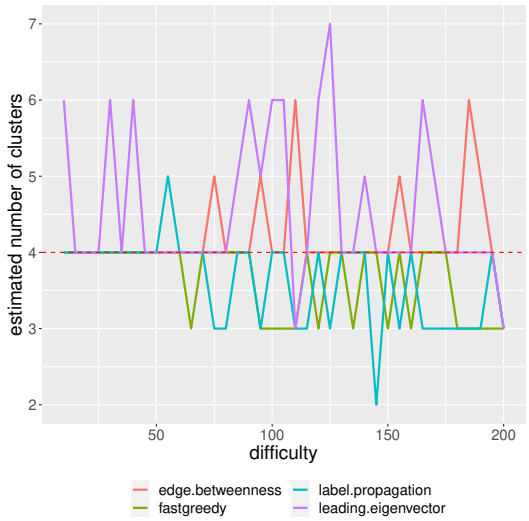

Figure 13: Estimated number of clusters for the **ER Network**, for increasing difficulty.
Real number of groups: 4.
Metrics: *edge betweenness*, *fastgreedy*, *label propagation*, *leading eigenvector*

Figure 14: Estimated number of clusters for the **ER Network**, for increasing difficulty.
Real number of groups: 4.
Metrics: *multilevel*, *spinglass*, *walktrap*, *infomap*

3.3 Algorithms

Out of the ten proposed algorithms, we only managed to implement nine, as **optimal.community** always resulted in a crush, even with the smallest of the Networks we considered. Concerning the **leading eigenvector** algorithm, we increased the maximum iterations parameter to a very high value to allow its convergence. Finally, in order to avoid issues with any of the algorithms, we preprocess each Network by removing the direction to its edges if present, and by applying simplification (hyperedges and loops removal).

3.4 Task 2 Decisions

Once having such a big graph, we had two simple solutions to overcome and face its analysis. First one, make a random selection and extract a smaller sample from the graph. With this option we could have applied a higher variety of clustering algorithms and have a richer comparison in the solution space. However, maybe we could have had some bias in the community detection depending on the sample procedure.

The second option, which we have chosen, consists of applying algorithms with lower costs to be able to analyze the whole space. Maybe is not that functional but we have thought that, as a lab exercise, it could be more interesting to overcome the problems produced by this choice rather than performing a similar work like task 1.

On the other hand, we have considered an undirected version of the network to have a bigger variety of clustering options but also, work with a simpler graph without losing much interpretive capacity of the results.

4 Discussion

4.1 General Results

For the four Networks, we determined the best community detection algorithm based on the scores over all 4 considered measures. Thus, to visualize their relative performance, we plot the values of *conductance*, *expansion*, *modularity*, and *tpr* for each algorithm in each Network: results are shown in **Figures 1, 3, 5, 7**. The first two measures are to be minimized, while the last two have to be maximised. The **label propagation** algorithm turns out to be the best for the *Macaque Network*, with 2 estimated clusters, while in the *Hospital Network* **walktrap** has the scores the best, identifying 3 different groups.

Concerning the two artificial Networks, **fastgreedy** is the best algorithm in terms of measures' values, but it fails to detect the 4 groups in both cases.

In **Figure 15** are shown the relative elapsed time required to detect communities by each of the analysed algorithms, for the four Networks. Considering the Networks summary statistics, displayed in **Table 6**, we can see how for the two smallest graphs, the algorithm with the highest elapsed time is **spinglass**. However, as network size increases, the computational requirements of **edge betweenness** lead it to become the slowest, taking up to 38 seconds to run.

Network	Nodes	Edges	Edge Density	Mean Degree
Macaque	45	255	5.67	11.33
Hospital	75	32424	432.32	864.64
Complete	234	8440	36.07	72.14
ER	234	5263	22.49	44.98

Table 6: Descriptive statistics of the considered Networks.

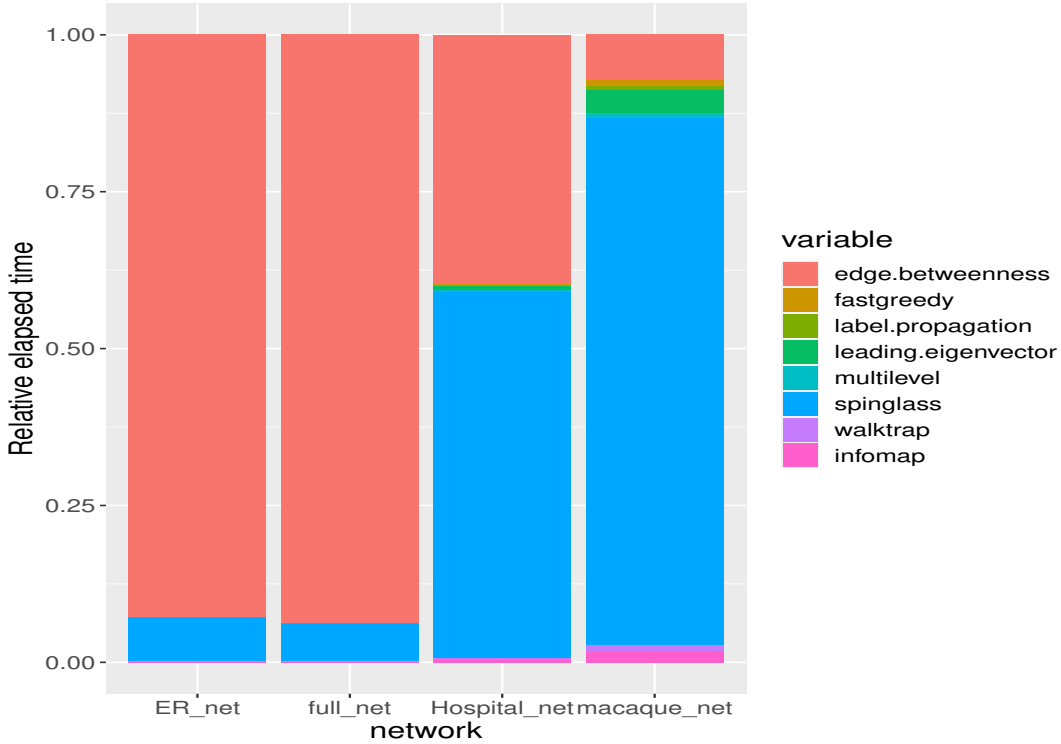

Figure 15: Relative elapsed time of the algorithms to detect communities in each Network.

4.2 Algorithms Performance

Now, we will proceed to analyze the performance of the nine algorithms in the community detection task, taking into account both the properties of the leveraged Networks and the metrics definitions.

4.2.1 Real Networks

We first consider the ***Macaque Network***, the smallest and more sparse one, with an edge density and a mean degree very low compared to the others. Even if **label propagation** turns out to be the best, **infomap** also does well in minimizing *conductance* and *expansion*, and shows a higher modularity. Meanwhile, the clustering configuration suggested by **edge betweenness** has the highest values in the two metrics to be minimized, and the lowest in those that are to be maximised.

However, for the ***Hospital Network*** these considerations are not valid anymore. Indeed, the two algorithms that performed best in the small sparse network, now fail to detect any cluster (i.e. they identify 1 "cluster"). Thus, we disregard them and take as the best scores those associated to **edge betweenness** and **walktrap**. The remaining algorithms show a range of *expansion* values from 15 to 20. These are extremely high values compared to those obtained in the other Networks, but also compared to the value of 2.40 scored by **walktrap**, which is indeed chosen as the best. Yet, as we can see in **Figure 4**, the resulting groups are highly unbalanced, with a cluster absorbing the majority of the nodes, and two small clusters composed of 6 and 2 nodes each.

4.2.2 Artificial Networks

In both the Networks we artificially created, **fastgreedy** outperforms the other algorithms. However, it is one of the few ones that fail to detect all the 4 real underlying communities. And, as we see from **Figure 8**, some of the nodes are assigned a different membership with respect to the original groups. While **label propagation** has a comparable performance in the *Complete Network*, in the *Er Network* its scores are aligned to those of the other algorithms.

In **Figures** from **13** to **14** we show the estimated number of clusters by each algorithm, for increasing difficulty. In the **Complete Network**, out of all, only **leading eigenvector** and **spinglass** fail in detecting the four communities. However, while the former begins to range between 4 and 5 groups after 250 new connections have been added, the latter shows a very high variability for intermediate difficulty values, with a pick at 8. Interestingly, for higher between-clusters connections the variability decreases; this could be a simple sampling effect caused by the grid search.

Things are different when considering the *ER Network*. When dealing with communities where the members are sparsely connected, almost every algorithm fails at detecting the 4 original groups. Yet, **walktrap** always predicts the correct number of clusters, for every difficulty level, while **infomap** and **multilevel** also perform well, only under or over estimating once each. Once again, **spinglass** and **leading eigenvector** show the highest variability, even for low difficulty values.

Thus, it seems like **walktrap** is the algorithm better able to grasp the underlying communities of a Network, even with weakly connected groups. On the other hand, **spinglass** and **leading eigenvector** largely overestimate the real number of clusters, even when communities are completely connected and the connections between clusters are relatively little, in particular with regard to **spinglass**.

5 Conclusions

In this work we evaluated the performance of nine algorithms on a community detection task in four different Networks. For the first two, the assessment was based on their scores of *triangle participation ratio*, *expansion*, conductance, and modularity. Then, to be able to compare the number of estimated clusters against a known truth, we created two artificial Networks, with 4 communities, more or less sparse. By linearly increasing the number of between-cluster connections most algorithms perform consistently well in the *Complete Network*, while only **walktrap**, always correctly estimates the number of communities also in the *ER* one. In both Networks, **spinglass** and **leading eigenvector** show very high variability, and mostly under or over estimate the clusters.

By assessing different Networks according to a range of measures, it is clear that the choice of the algorithm should be strictly related to the properties of analysed graph. Moreover, one should put care in the implementation of the algorithms based on greedy search on modularity, if singletones are present. In conclusion, when facing big graphs, since visual conclusions are limited to small networks, metrics are an important tool to evaluate performance.