Complex Networks - Homework 3

Group 1: Luna Giesselbach, Iwan Pasveer, Flavia Leotta, Noah Frinking, Dylan Gavron

1st October 2024

The Python code for these exercises can be found on the Markdown file (*"Homework_03_group_1.ipynb"*) we provided as additional material.

Homework 3.1

The Python function

$$SBM(c, nc, p\_intra, p\_inter) \tag{1}$$

is a **Stochastic Block Model** function that creates a random graph with c communities, each of which has nc nodes that are connected within each other with a probability p_intra, and between nodes of different communities with a probability p_inter. Since, as we know from Chapter 3 of the Lecture notes, in SBM the nodes belonging to the same community are connected with higher probability than the nodes belonging to different communities, we also implemented an Exception that warns the user when an inter-probability bigger than the intra-probability is given.

We illustrated an example network (G_example_1) (*Figure* 1) and we also created a network called G1 with the following parameters: $c = 10$, $nc = 100$, $p\_intra = 0.1$, $p\_inter = 0.01$. We're not going to plot this but we're going to use this graph later.

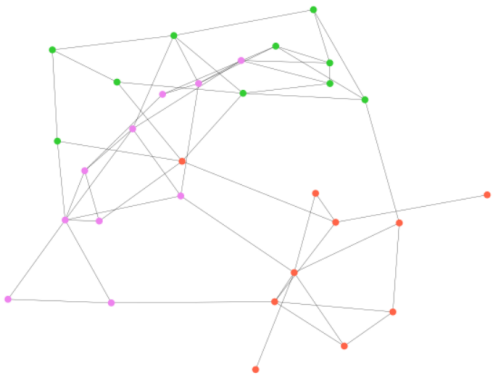

Figure 1: Example of graph G_example_1 created with function (1) (parameters: $c = 3$, $nc = 10$, $p\_intra = 0.2$ and $p\_inter = 0.05$) and plotted with $plot\_SBM()$ function. Three communities were created and coloured in red, pink and green, respectively.

Homework 3.2

We implemented an updated version of the SBM model, where all communities are of different sizes, provided in a list:

$$SBM\_D(comm\_size, p\_intra, p\_inter) \tag{2}$$

We illustrated an example ($G\_example\_2$) with $comm\_size = [25, 10, 5]$, $p\_intra = 0.2$ and $p\_inter = 0.05$, as shown in *Figure* 2.

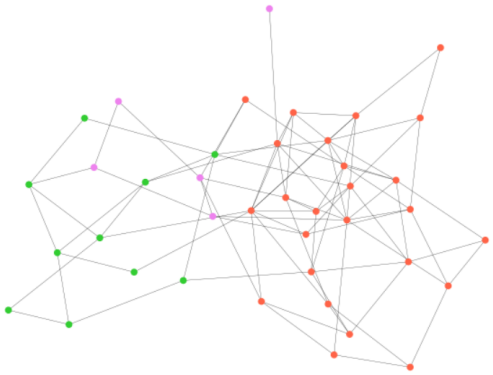

Figure 2: Example of graph $G\_example\_2$ created with function (2) (parameters: $comm\_size = [25, 10, 5]$, $p\_intra = 0.2$ and $p\_inter = 0.05$) and plotted with $plot\_SBM\_D()$ function. Three communities of different sizes were created: it can be seen that the red one is the biggest community (of size 25) while the pink one is the smallest (of size 5).

Using the same function we create two networks: G2 (parameters: ([500, 200, 100, 100, 100], 0.1, 0.01)) and G3 (parameters: ([600, 200, 100, 50, 50], 0.1, 0.01)). Both G2 and G3 aren't plotted, but are used in the next exercises.

Homework 3.3

We implemented the function

$$degree\_distribution(*graphs) \tag{3}$$

that plots the degree (k) distribution of different graphs in a single figure on log-log scale. We decided to allow this function to take up to 12 arguments, potentially comparing more than 3 graphs in one figure. In our example (*Figure* 3), we showed the comparison between G1, G2 and G3 (created in Homework 3.1 and 3.2).

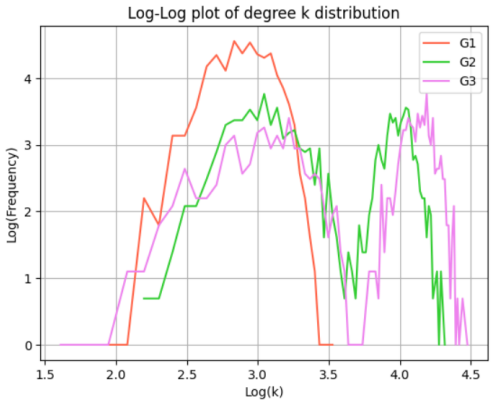

Figure 3: Plot of the degree (k) distribution of graphs G1, G2 and G3 using function (3).

To better explain the behaviour of the plot, we summarised the parameters used for the creation of the three graphs in *Table* 1. The presence of differently placed peaks in *Figure* 3 shows that we're

comparing graphs that share both similarities and differences. Even though they are composed of the same total number of nodes (1000) and share the same p_intra and p_inter, they contain communities of much different sizes.

Graph	Communities' size	Tot nodes	p_intra	p_inter
G1	100, 100, 100, 100, 100, 100, 100, 100, 100, 100	1000	0.1	0.01
G2	500, 200, 100, 100, 100	1000	0.1	0.01
G3	600, 200, 100, 50, 50	1000	0.1	0.01

Table 1: Parameters of graphs G1, G2 and G3.

As shown in *Table* 1, G1 has 10 communities of all the same size so there's only one peak which only depends on the size of the communities (100 nodes) and the p_intra set (0.1). But it's possible to notice that, in the same position, both G2 and G3 show a peak, even if smaller in size. This is because G2 and G3 both have communities of size 100 and same p_intra as G1.

But differently from G1, G2 and G3 have also communities of different size: G2's biggest one contains 500 nodes and the smallest 100 (like all the G1 communities) and G3 has an even more extreme situation with the biggest community containing 600 nodes and the smallest only 50. For this reason G1 and G2 both show two peaks.

The two peaks, though, look quite different: G3 has the first peak spreading more towards a smaller degree k and the second peak being taller and shifted towards higher degrees. This is because G3 communities of 50 nodes, even though they have the same probability to form edges as other communities, will potentially form less of them due to containing fewer "potential candidates" (nodes). The same explanation can be given about the second peak being shifted towards higher degree k, since the G3 community of 600 nodes have more "potential candidates" so an overall higher k.

Homework 3.4

We implemented the function

$$core\_number\_distribution(*graphs) \tag{4}$$

that plots the core-number distribution of graphs (this function takes a maximum of 12 arguments in our code). We applied it to the graphs G1, G2 and G3, obtaining the plot in *Figure* 4.

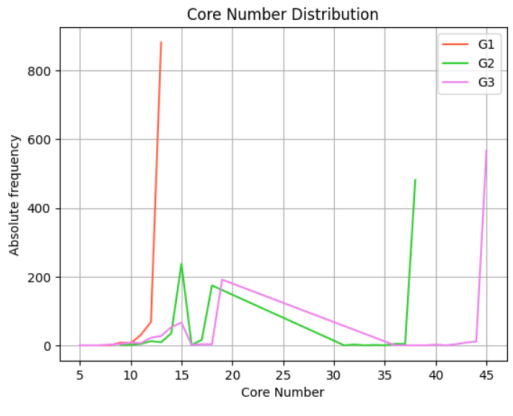

Figure 4: Plot of the distribution of core numbers in graphs G1, G2 and G3, using function (4).

A k-core is a maximal subgraph that contains nodes of degree k or more, and the core number of a node is the largest value k of a k-core containing that node. So this means that, the higher the number of degrees of a node, the more likely the degree is to belong to a k-core with a high value of k, and thus the higher the core number will be.

G1 has communities that all have the same amount of nodes with the same probability of connecting to each other and to nodes of different communities. The core number is thus roughly equal for all nodes, as all nodes have the same probability to connecting to the same amount of other nodes and thus will have roughly the same amount of connections.

In G2, 3 of the 5 communities have the same size. Since the probability to connect within the community is larger than to connect outside of the community, we're going to take more into consideration the community size, because it's what affects the core number the most. In the 3 smaller communities, all of the 100 nodes will have less connections as there are fewer nodes within the community to connect to, compared to the larger communities. The first peak around 14 can be explained by the 3 smaller communities of 100 nodes. This number is easily explained by statistics: the probability of creating an edge in the same community is 0.1, so each node is, on average, going to be connected to around the 10% of the nodes in the same community. Plus, in smaller amount, they're also going to be connected to some of the nodes from the other communities. The largest peak in G2 comes from the largest community of 500 nodes, as these nodes have the largest average degree and thus the largest core number. The other peak around 18 comes from the community with size 200.

For G3, we see the same peak around 14 as for G2, coming from the community with size 100. The peak is almost half the size of the peak of graph 2, as graph 2 has twice as much (two) communities with 100 nodes. The peak around 18 is also comparable to the peak from graph 2, as G3 also has a community with 200 nodes. The tallest peak of G3, with a much higher core number than the largest peak of G2, comes from the community with 600 nodes and is around 10 units of core number higher than the G2 peak (for the community with 500 nodes). This is because the probability to connect with each node within a community is 10% and the G3 community of 600 nodes has, on average, a degree k 10 units higher per node, than the G2 community of 500 nodes. The smaller communities don't influence much the core number, as, not only there is just a 1% chance to connect with them, but also their number of nodes is too small in comparison to the bigger communities.

Homework 3.5

We implemented the function

$$compare\_communities(*graphs) \tag{5}$$

that applies different community detection methods (Greedy, Label propagation and Louvain) and then prints their modularity. The results are reported in *Table* 2.

Graph	Greedy	Label-propagation	Louvain
G1	0.324	0.000	0.427
G2	0.280	0.164	0.293
G3	0.191	0.131	0.211

Table 2: Modularity of G1, G2 and G3 graphs, using different community detection methods, by applying function (5).

We observed that both the Louvain and Greedy algorithms perform relatively well in identifying community structures (higher modularity), but the Label Propagation algorithm shows a unique behavior in G1. Modularity can be defined as:

$$Q = \sum_{c=1}^{n} \left[\frac{L_c}{m} - \gamma \left(\frac{k_c}{2m} \right)^2 \right] \text{[1]} \tag{6}$$

with L_c being the number of intra-community links for community c, k_c the sum of the degrees of nodes in community c and m the number of edges. We will use resolution parameter γ set equal to 1, as it's commonly used [2].

Modularity measures how well a network is partitioned into communities by comparing the actual density of intra-community links ($\frac{L_c}{m}$) to the expected density if the links were placed randomly ($\frac{k_c}{2m}$).

So the larger the sum of degrees k_c is compared to the intra-community links L_c, the more links there are outside of the community (inter-community links) and the smaller the result of (6). It follows that, the lower the modularity, the more links there are that are inter-community with respect to the intra-community links.

Each method works slightly differently:

- The Greedy algorithm finds communities using greedy modularity maximization. It begins with each node as its own community and iteratively merges communities to increase the overall modularity by choosing the move that maximizes the modularity gain. An important feature of greedy selection is that it does not backtrack. [3]
- In the label-propagation algorithm every node is initialized with a label and subsequently it's updated to match it with the most common label of its neighbours. The procedure continues until a stable set of labels is reached, and nodes with the same label form a community. [4]
- The Louvain algorithm starts by assigning a different community to each node of the network (so there are as many communities as there are nodes). Then, for each node, it considers the neighbours of it and evaluates the gain of modularity that would take place by removing the node from its community and by placing it in the community of the neighbour. The node is then placed in the community for which this gain is maximum and if no positive gain is possible, it stays in its original community. This process is applied until no further improvement can be achieved. The second phase involves aggregating communities and optimizing again. [5]

We can see that, in all the three graphs, the Louvain algorithm is more successful in identifying the communities than the greedy algorithm. This is the case because it works on multiple levels, first locally and then more globally. This strategy is more efficient in identifying the communities because it lowers the risks of the algorithm getting 'stuck' on a local best solution and not finding the global one.

Both algorithms give a higher modularity when the community sizes are more evenly distributed, as from G1 to G3 we can notice a gradual worsening in the results. In graphs where there is a significantly bigger community with highly connected nodes (G2 and G3), the algorithms choose (falsely) to add these nodes to its newly formed community, because of the inter-connections with this big community might overrule the intra-connections in the smaller communities.

The label-propagation algorithm provides a different result: the modularity is 0 when used on G1. This may be the case because all the nodes in G1 are on average evenly connected with 10 communities of the same size: the uniformity of community sizes in G1 does not provide enough distinguishing characteristics for the labels to propagate effectively. G1 begins with the stable set of labels that the algorithm looks for. Differently, the varying sizes in G2 and G3 help the algorithm in identifying distinct communities. In G2 and G3 the chance of a group of nodes of the same community having the same label assigned increases. After the big cluster is identified as a community the rest of the graph is also better and more easily divided into other communities.

But we also want to understand the reason why the label propagation is worse in the case of G3, where the communities' sizes differ more. As we stated, the Label-Propagation method works the best if there are clear distinctions between communities, allowing for effective label convergence. But, if the largest community in G3 is more strongly connected to the smaller communities (higher inter-community connections) than the smaller communities' nodes are connected to each other (lower intra-community connections), the algorithm can still struggle to differentiate these communities. So the smaller communities's nodes might share more neighbors with the bigger community than with each other, preventing distinct label propagation, although it is not as pronounced as in G1.

Additional Material

This report is accompanied by a Markdown file (*"Homework_03_group_1.ipynb"*) which contains the Python code. Several libraries are used, and their version stated in the list below.

- Python version: 3.12.6
- NetworkX version: 3.3
- Matplotlib version: 3.9.0
- Community version: 0.16

References

[1] Clauset, Aaron, Mark EJ Newman, and Cristopher Moore. "Finding community structure in very large networks." *Phys. Rev.* E 70.6 (2004). https://arxiv.org/abs/cond-mat/0408187.

[2] M. E. J. Newman, "Equivalence between modularity optimization and maximum likelihood methods for community detection" *Phys. Rev.* E 94, 052315, 2016. doi.org/10.1103/PhysRevE.94.052315.

[3] Wang, Yizhun. (2023). Review on greedy algorithm. *Theoretical and Natural Science*. 14. 233-239. 10.54254/2753-8818/14/20241041.

[4] Brahim Laassem, Ali Idarrou, Loubna Boujlaleb, M'bark Iggane, "Label propagation algorithm for community detection based on Coulomb's law", *Physica A: Statistical Mechanics and its Applications*, Volume 593, 2022, 126881, ISSN 0378-4371, doi.org/10.1016/j.physa.2022.126881.

[5] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte and Etienne Lefebvre. "Fast unfolding of communities in large networks", *Journal of Statistical Mechanics: Theory and Experiment*, Volume 2008, 2008, 10.1088/1742-5468/2008/10/P10008