UNIVERSITAT ROVIRA I VIRGILI
Department of Computer Engineering and Mathematics
April 15, 2021

CN-MAI Assignment 3

Complex Networks
Spring 2021, 5 Credits

Community Detection

Name Bartosz Paulewicz
Betty Törnkvist

E-mail bartosz.paulewicz@student.put.poznan.pl
betty.tornkvist@gmail.com

Teacher
Alejandro Arenas Moreno

Contents

1 Community Detection Algorithms

When examining network structures, densely connected nodes can be discovered and defined as communities. The practical implications are many: social groups with similar interests can be detected, as well as manipulative groups such as stock market tamperers, etc. In comparison to typical machine-learning clustering, community detection algorithms are tailored for complex networks and takes advantage of the information related to edges.

In the following sections we took a look at three different community detection algorithms, all based on optimization of modularity which evaluates partitions of a network into communities by calculating relative density of edges within communities compared to edges outside communities. It means that the nodes in the networks with high modularity are densely connected within their community and sparsely connected to nodes outside their community.

1.1 Fast-Greedy

The fast-greedy algorithm[1], proposed in 2004, aims to maximize the difference between modularity with current communities and modularity after changing assigned community of one of the nodes. It begins with each node assigned to its own community. Then for each node the difference in modularity is computed for moving that node from its current community to the community of each of its neighbors. Once the modularity differences are computed for all its neighbours the node is moved to the community where the modularity gain is the highest. If no move can increase modularity leave node in its initial community. This is sequentially repeated for all nodes until no modularity gain is possible.[2] The time complexity of this algorithm is $|V||E|\log(|V|)$.

1.2 Louvain

The Louvain algorithm[3] (also called multi-level) was introduced in 2008 as a way to detect communities in large networks. It is also modularity-based and aims to maximise the modularity by using heuristics and repeating two steps - moving nodes locally and aggregate the network. The first step starts by assigning nodes to communities. The modularity is then measured when a node is moved to another community, and if the modularity is improved, the node is moved. When a local maximum is reached and a hierarchy has been created, the first step is interrupted. In the second step, a new network is constructed where the new communities are considered nodes. The steps are repeated until the network does not change any more and the highest modularity is achieved.[4] This algorithm seems to run in near-linear time in the number of edges but for more challenging graphs the runtime of the this algorithm almost becomes quadratic in the number of nodes.

For the Louvain-implementation, the library community_louvain was used.

1.3 Leiden

The Leiden algorithm[5] is an improvement of the Louvain algorithm, which was presented in 2019. They showed that the Louvain algorithm sometimes creates communities of nodes that are badly connected, or not connected at all.

To approach this issue and reassure well-connected networks, the Leiden method adds a step in between the two: refinement of partitions. The proposed communities in step one can be split into further partitions. By randomly connecting nodes, a larger space of partitions can be examined. The Leiden method also tweaks with the first step by only revisiting nodes that have changed communities in the previous iteration.[4] This algorithm seems to run in near-linear time in the number of edges even for more challenging graphs the runtime of the this algorithm almost becomes quadratic in the number of nodes.

For the Leiden-implementation, the library leidenalg was used.

2 Results

Previously mentioned methods were applied to a provided data set containing `.net` networks and their reference partition in `.clu` format which was provided from external resources.

2.1 Modularity

Since all detection methods are modularity-based, the modularity for all networks were calculated, and can be seen in Table 1.

Modularity				
	Fast-Greedy	**Louvain**	**Leiden**	**Reference**
cat_cortex_sim	0.260436	0.266097	0.266097	0.245996
dolphins	0.495491	0.523338	0.520114	0.373482
zachary	0.380671	0.419790	0.419790	0.371466
football	0.549741	0.604407	0.597230	0.553973

Table 1: Modularity for the different methods, calculated for each given network.

2.2 Comparison Metrics

Three different metrics were calculated to measure the outcome of the community detection methods: Jaccard Index, Normalized Mutual Information and Normalized Variation of Information.

All of those metrics are symmetric which means that for comparing communities X and Y switching X and Y will produce the same score value. This can be useful to measure the agreement of two independent label assignments strategies on the same network even when the real ground truth is not known.

2.2.1 Jaccard Index

The Jaccard Index (JI) is defined as the size of the intersection divided by the size of the union of two label sets and is used to compare set of predicted labels to the corresponding set of ground truth labels. The higher the score ([0,1]), the more similar the sets are. We used a weighted average, which means that JI is calculated for each community and then averaged, see Table 2.

$$JI(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}$$

JI is dependent on the absolute values of the labels which means that, for example, adding one to each label value in one of the communities can change the score value. Therefore this makes it poor metric for evaluating similarity between communities. The JI for the different networks and algorithms can be seen in Table 2.

Jaccard Index			
	Fast-Greedy	**Louvain**	**Leiden**
cat_cortex_sim	0.515731	0.540355	0.017833
dolphins	0.664223	0.398094	0.274194
zachary	0.255656	0.323529	0.000000
football	0.123395	0.200000	0.104348

Table 2: Jaccard Index for the methods, calculated for each given network.

2.2.2 Normalized Mutual Information

The Mutual Information is a measure of the similarity between two communities. Where $|X_i|$ is the number of the samples in community X_i and $|Y_j|$ is the number of the samples in community Y_j, the Mutual Information between communities X and Y is given as:

$$MI(X,Y) = \sum_{i=1}^{|X|}\sum_{j=1}^{|Y|}\frac{|X_i \cap Y_j|}{N} log\frac{N|X_i \cap Y_j|}{|X_i||Y_j|}$$

This metrics is independent of the absolute values of the labels which means that a permutation of the label values won't change the score value. Therefore this makes it good metric for evaluating similarity between communities.

Normalized Mutual Information is a normalization of the Mutual Information score to scale the results between 0 (no mutual information) and 1 (perfect correlation). The NMI for the different networks and algorithms can be seen in Table 3.

Normalized Mutual Information			
	Fast-Greedy	**Louvain**	**Leiden**
cat_cortex_sim	0.656873	0.672641	0.672641
dolphins	0.572700	0.513889	0.455796
zachary	0.692467	0.687263	0.687263
football	0.697732	0.856083	0.848707

Table 3: Normalized Mutual Information for the methods, calculated for each given network.

2.2.3 Normalized Variation of Information

Variation of Information is a measure of the distance between two communities closely related to mutual information. Unlike mutual information, the variation of information is a true metric, in that it obeys the triangle inequality. Suppose we have two partitions X and Y of a set A into disjoint subsets:

$$X = \{X_1, X_2, ..., X_k\} \quad \text{and} \quad Y = \{Y_1, Y_2, ..., Y_l\}$$

$$n = \sum_{i}^{X}|X_i| = \sum_{j}^{Y}|Y_j| = |A|$$

$$p_i = \frac{|X_i|}{n} \quad \text{and} \quad q_j = \frac{|Y_j|}{n}$$

$$r_{ij} = \frac{|X_i \cap Y_j|}{n}$$

Then the variation of information between the two communities is:

$$VI(X,Y) = -\sum_{i=1}^{|X|}\sum_{j=1}^{|Y|} r_{ij}(log\frac{r_{ij}}{p_i} + log\frac{r_{ij}}{q_j})$$

This metrics is independent of the absolute values of the labels which means that a permutation of the label values won't change the score value. Therefore this makes it good metric for evaluating similarity between communities.

Normalized Variation of Information is a normalization of the Variation of Information to scale it between 0 (identical communities) and 1 (totally different communities) done by dividing it by the maximum possible value which is $log(n)$. The NVI for the different networks and algorithms can be seen in Table 4.

Normalized Variation of Information			
	Fast-Greedy	**Louvain**	**Leiden**
cat_cortex_sim	0.142503	0.137325	0.137325
dolphins	0.130497	0.175363	0.199315
zachary	0.532115	0.856328	0.629254
football	0.185393	0.096596	0.101359

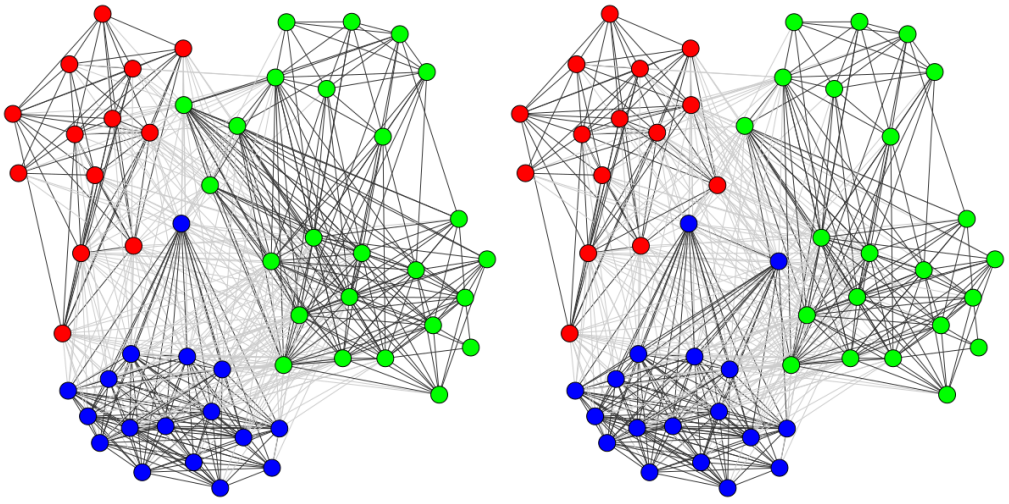
Table 4: Normalized Variation of Information for the methods, calculated for each given network.

2.3 Community Plots

All the plots of the different partitions of the same network were grouped by network and put together. The plots are color coded and the layout is automatically chosen to visualize the best option, see Figure 1-4.

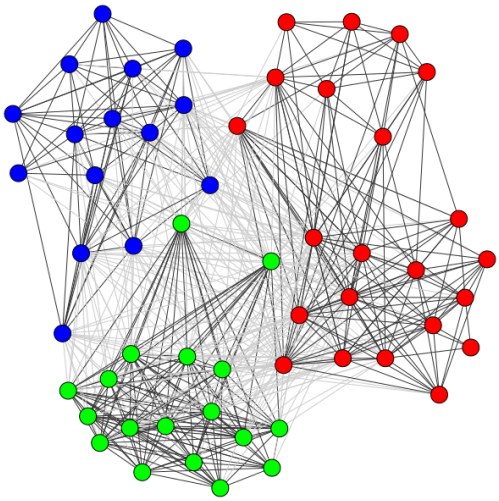
To decide on a specific graph layout we used function layout_auto from igraph library. This function automatically chooses most appropriate layout based on topological properties of the graph using the following rules:

1. If the graph has vertex attributes called x and y, these will be used as coordinates in the layout. (airports, cat_cortex_sim and zachary)
2. Otherwise, if the graph is connected and has at most 100 vertices, the Kamada-Kawai layout will be used. (dolphins)
3. Otherwise, if the graph has at most 1000 vertices, the Fruchterman-Reingold layout will be used. (football)
4. If everything else above failed, the DrL layout algorithm will be used. (none in our case)

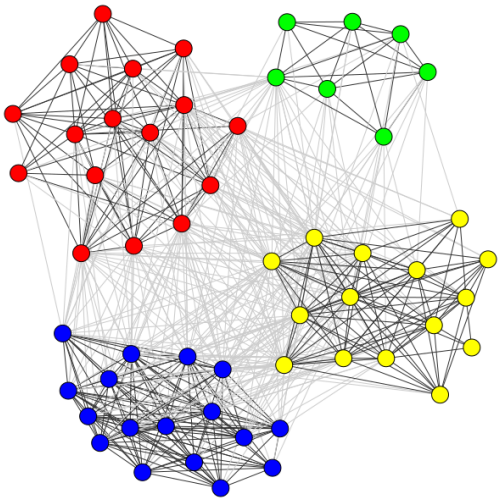


(a) Fast-Greedy.

(b) Louvain.

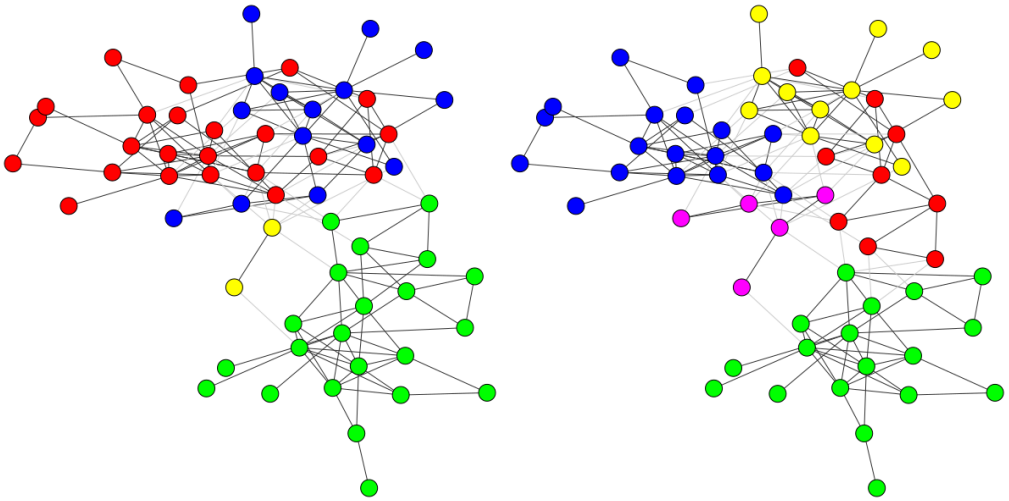


(c) Leiden.



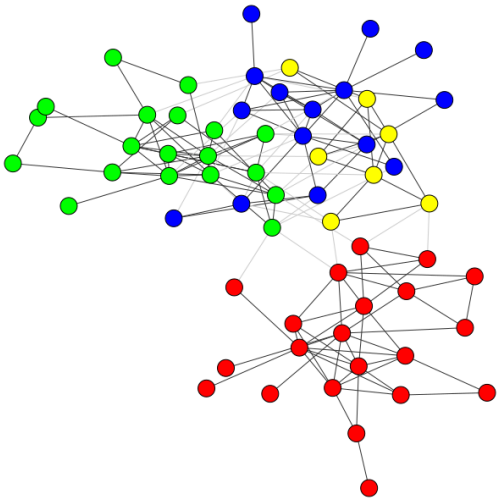
(d) Reference.

Figure 1: Community detection for the network cat_cortex_sim.

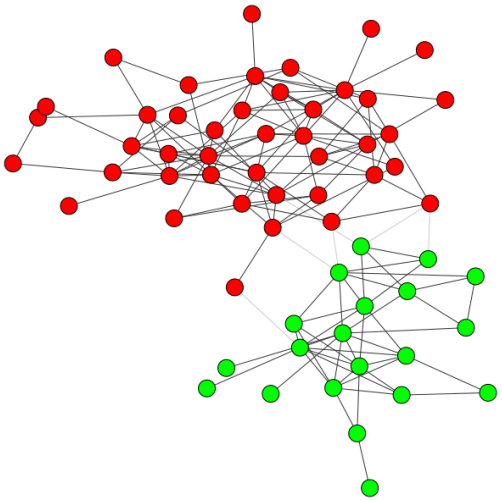


(a) Fast-Greedy.

(b) Louvain.

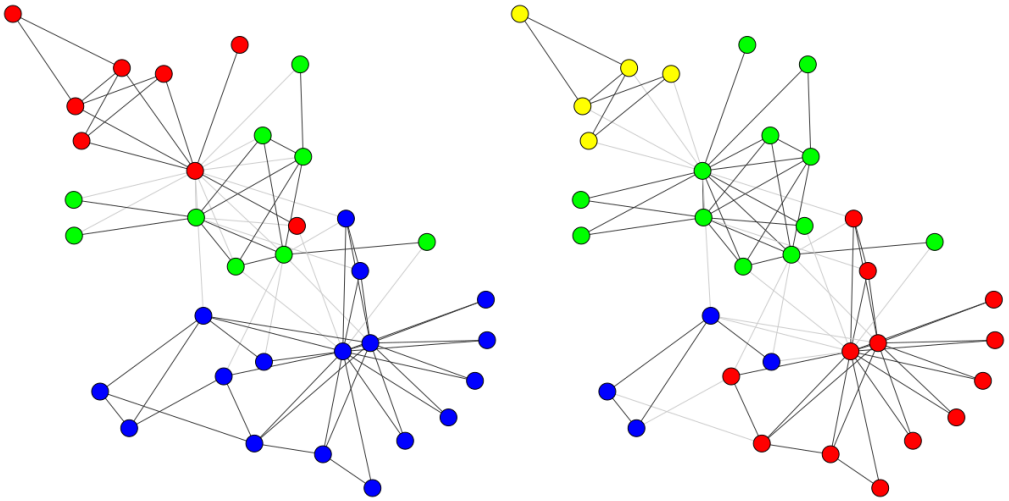


(c) Leiden.



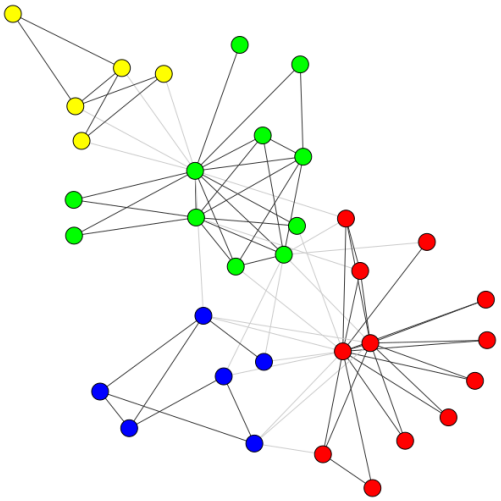
(d) Reference.

Figure 2: Community detection for the network dolphins.

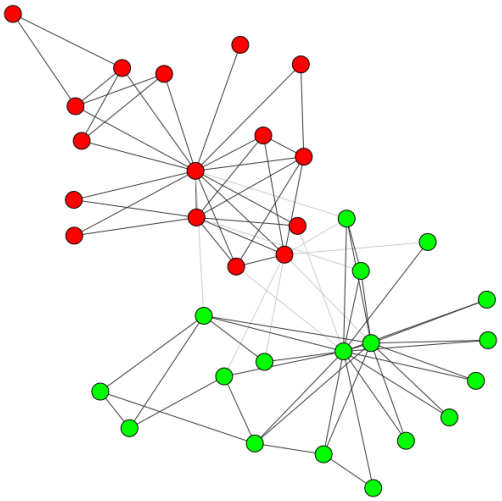


(a) Fast-Greedy.

(b) Louvain.

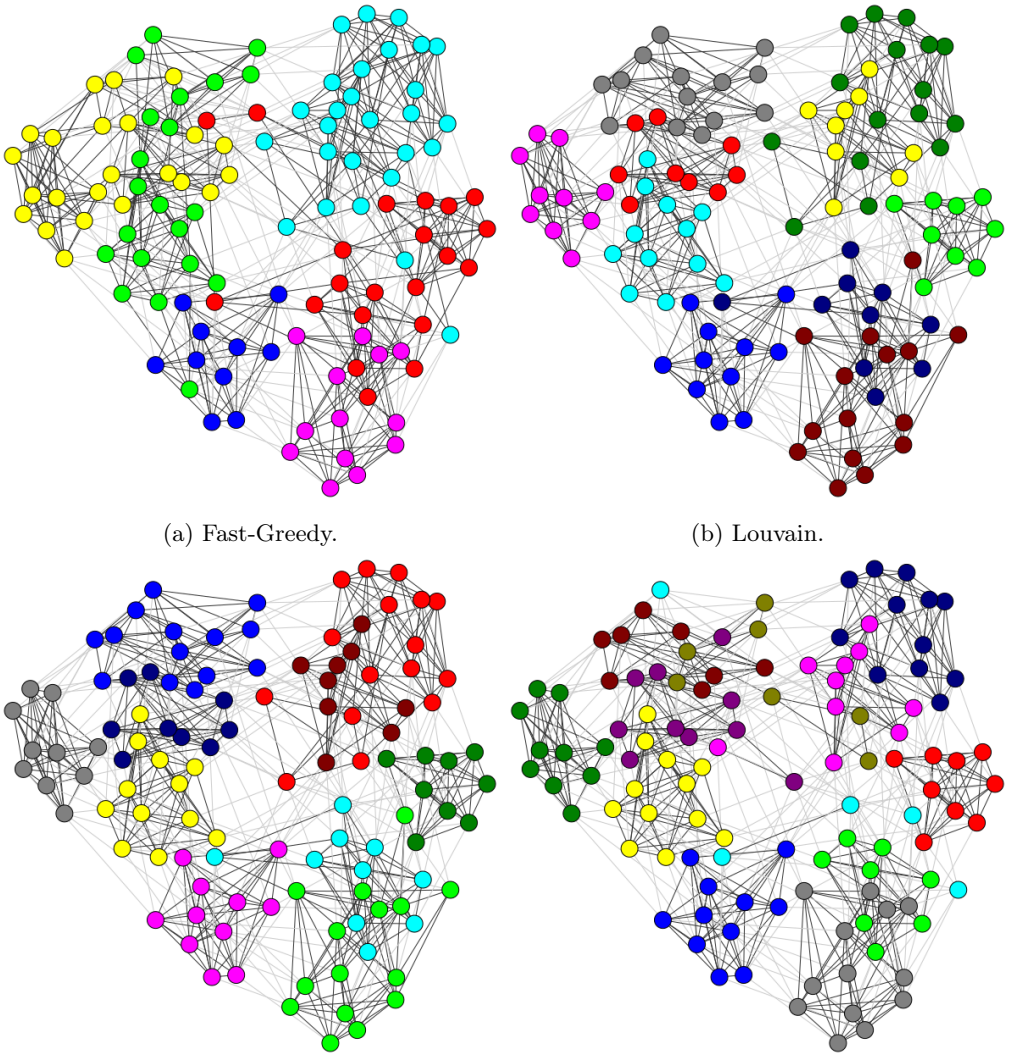


(c) Leiden.



(d) Reference.

Figure 3: Community detection for the network zachary.



(a) Fast-Greedy. (b) Louvain.

(c) Leiden. (d) Reference.

Figure 4: Community detection for the network football.

3 Conclusion

When comparing the three methods, we can observe that they have their different strengths and weaknesses, see Table 5. They all have a limitation in common - the resolution limit. Since they all are built for modularity optimization, there is a chance that smaller communities that are otherwise well defined as separate, are merged together into a larger community. Leiden, which has been proposed in recent years has a more refined method in many aspects to the others. By fixing the flaws of Louvains weakly connected nodes and using a random approach instead of a greedy, Leiden should result in better communities than the other methods. In the first plot, see

	Advantage	**Limitation**
Fast-Greedy	Fast, easy implementation	Can have issues detecting smaller well defined communities in larger networks due to the resolution limit.
Louvain	Finds communities within communities, easy implementation.	Can discover weakly connected/disconnected communities and can be slow since it revisits nodes that has not changed neighbours. Stores network in main memory. Can have issues detecting smaller well defined communities in larger networks due to the resolution limit.
Leiden	Faster than Louvain, only re-visits nodes that has recently been assigned a new neighbour. Guarantees well-connected communities. Non-greedy allows broader discovery of the partition space.	Can have issues detecting smaller well defined communities in larger networks due to the resolution limit.

Table 5: Advantages and drawbacks of the different community detection algorithms.

Figure 1, Louvain and Leiden are identical, while fast-greedy only differs slightly. The reference network shows a fourth community that neither of the methods where able to detect, and this might be a result of the resolution limit. However, in the second plot, see Figure 2, the Louvain-plot has detected five communities, while the reference only presents two. Fast-greedy has created a community with only two nodes. All three plots seems to try to identify communities in the upper cluster, which visually looks a bit overlapping. In the third plot shows similar results in terms of number of communities detected, see Figure 3. Again, Louvain and Leiden are very similar, whilst the fast-greedy plot does not separate the communities as well. Finally, in Figure 4, we can again observe the similarity of the Louvain and Leiden methods, which is also more similar to the given reference. However, it should be noted that the reference is an unknown partition and does not necessarily represent the ground truth, and might therefore be deceiving in ways of comparison.

The Jaccard Index measures in Table 2 shows the poorest results for the Leiden-method. However, as mentioned previously, the JI depends on the absolute values of the labels which makes it a poor metric to compare how similar the communities are. The Normalized Mutual Information does not suffer this flaw, however the calculated values does not show any larger variance in Table 3, where the results for Leiden is identical or slightly worse than Louvain and Fast-greedy. For the Normalized Variation of Information in Table 4, it can be observed that Leiden and Louvain created more non-identical communities with an exception for the final network (football).

Our results shows no clear advantages to the Leiden method, which would otherwise be expected from a theoretical viewpoint. However, this might be due to the small number of networks in our experiment and their size, where all networks has quite few nodes (<150). A pattern of similarity between Louvain and Leiden is clear, which makes sense considering that Leiden developed from Louvain.

References

[1] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review E*, vol. 69, no. 6, p. 066133, 2004.

[2] L. Rita. Modularity maximization. [Online]. Available: https://towardsdatascience.com/modularity-maximization-5cfa6495b286

[3] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[4] T. D. Jayawickrama. Community detection algorithms. [Online]. Available: https://towardsdatascience.com/community-detection-algorithms-9bd8951e7dae

[5] V. A. Traag, L. Waltman, and N. J. Van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.