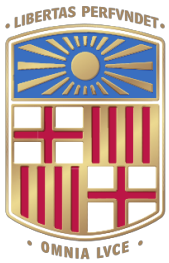


Practical Exercise 3: Community Detection in Complex Networks

Complex Networks

Master in Artificial Intelligence

Students:
- Alam López
- Mario Rosas

Professors:
-Alex Arenas
-Sergio Gómez

Academic Year: 2023-2

Laboratory Group: 1

April 14, 2024

Contents

Community Detection

1 | Introduction

Understanding the structure of complex networks is fundamental in various fields, from social sciences to biology, among others. Community detection algorithms serve as essential tools to find the organization of such networks according to their properties, from which we can identify connectivity patterns that reveal communities within them where nodes take a particular distribution tending to cluster together. In this practical work, we experimented in several ways to find the community structures within synthetic and real world networks, this allows us to know and understand the algorithms for community detection, as well as the algorithms for performance evaluation of the same when the true labels are known, we could also understand how the variation of probabilities in terms of connectivity influence the results of such algorithms.

By analyzing synthetic networks generated through the stochastic block model (SBM) and a real-world network capturing interactions in a school environment, we measured the effectiveness and observed the limitations of different algorithms in delineating community structures.

Synthetic networks, parameterized by the probabilities of intra- and interblock connections, provide a controlled environment for evaluating the performance of community detection algorithms. Using networks with systematic variations of this parameter, we experiment and observe the ability of the algorithms to adapt to different degrees of community cohesion and the accuracy of their partitions compared to the ground truth.

In addition, we generate visualizations of the community structures of networks with different connection probabilities, and this provides us with information about the clustering tendencies of the algorithms visually and helps us understand their robustness in capturing the underlying clusters. In the second exercise analyzing an interaction network in an elementary school we discover the community structures between students and teachers. By examining weighted and unweighted versions of the network, it is possible to validate the impact of interaction intensity on community detection results and explore how these structures align with those provided by existing data within the school.

2 | Characterization of the community structure of networks with block structure

2.1 | Task Description

This task focuses on characterizing the community structure of synthetic block-structured networks using community detection algorithms. These networks are generated based on the stochastic block model (SBM), with parameters including the number of nodes (N), the number of different blocks (nblocks), and the probabilities of intra- and inter-block connections (prr and prs, respectively). With N set to 300 and $nblocks$ to 5, and prs to 0.02 we varied the prr parameter from 0 to 1 to examine the impact of different degrees of connectivity within a block on community detection.

To accomplish this task, we use the implementation of 4 community detection algorithms: Infomap, Agglomerative Greedy, Louvain and Leiden, the last 3 being of modularity maximization. These algorithms are responsible for identifying cohesive groups of nodes within synthetic networks, with known ground truth clusters for evaluation. By analyzing the evolution of the number of communities and modularity as prr changes, we seek to understand how different algorithms respond to different levels of connectivity within a block and their ability to accurately detect community structures. In addition, we evaluate the performance of each algorithm by comparing the partitions they identify to the ground truth using the Jaccard index, normalized mutual information, and normalized information variance. Finally, we visually represented the community structures of the networks with different prr values to facilitate comparison and visual analysis of the performance of the community detection algorithms, using an special algorithm to locate the nodes and used color coding to indicate the community assignments of each algorithm.

2.2 | Implementation

Unlike the previous assignments that we programmed in Julia, we programmed this one in Python. By moving from Julia to Python for this assignment, we took advantage of the flexibility and robustness of the libraries available in Python, particularly for the community detection algorithms. Our main implementation, located in the *NetworksCommunityDetection.py* script, contains the functions that enable the analysis of synthetic networks with block structure.

The implementation starts with a function to load networks. This is followed by community detection that allows the selection of one of several methods, such as Louvain, Infomap, Leiden and Agglomerative clustering. The implementations of the Louvain, Infomap and Leiden algorithms were taken from different Python libraries that have such algorithm pre-implemented. For the Aggolmer-

ative Greedy algorithm, functions were used to generate the hierarchical element linking process in order to consolidate the partitions, the linking was performed under the ward criterion, which minimizes the variance of the closters that are joined. For clustering, distance was used as the criterion for joining groups, which forms flat clusters so that the original observations in each flat cluster have no greater a cophenetic distance than a given parameter.

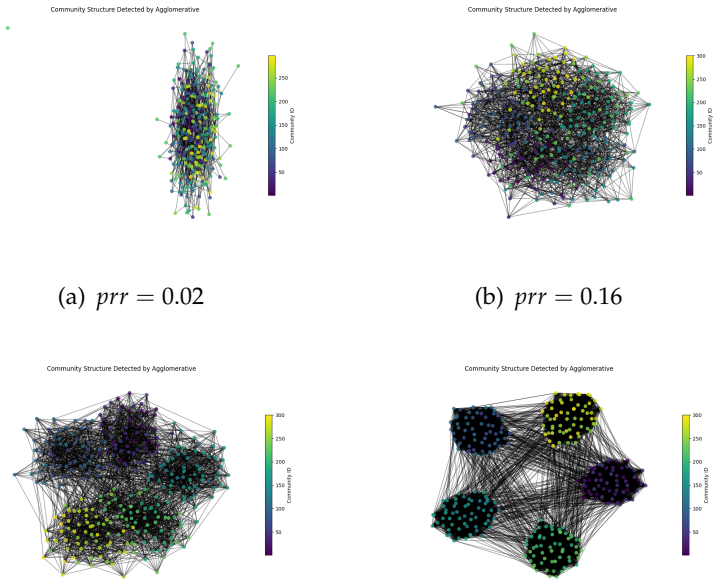
To evaluate the performance of each algorithm, we calculated Normalized mutual information (NMI), Normalized variation of information (NVI) and Jaccard index metrics. These metrics were partially taken from existing Python libraries, but NVI and Jaccard index were adapted to be compatible with the community detection task and not give misleading results.

In addition, we visualised the detected community structures using matplotlib, generating colour-coded representations of the community assignments of the networks. For the node arrangement in the visualisation we used the implementation of the Fruchterman-Reingold force-directed algorithm provided by one of the python libraries for network management. Finally, the implementation includes a script, *CommunityDetectionTests.py*, to perform automatic tests on multiple synthetic networks and aggregate the results into a complete CSV file, streamlining the analysis process and allowing systematic comparison between algorithms and network configurations.

2.3 | Results

Using the information in figure 3.1 it is possible to see that the impact of increasing intra-block connectivity probability (prr) on community detection using an Agglomerative Greedy Algorithm. At a low prr (0.02), community structures are sparse and poorly defined, there is also a node without any connection causing the layout algorithm to position the entire compressed network on the right-hand side. As prr rises to 0.16 there is a better separation of the groups, however, there is still a clear interposition between groups. When prr is set to 0.28, communities become more compact and discernible, indicating that higher intra-community links aid the algorithm's effectiveness. With prr at 1.00, each community is densely interconnected, showcasing the algorithm's optimal performance in identifying clear and distinct community structures, although it is still possible to see multiple variations of colour shades within each defined group.

As shown in figure 3.2, Leiden's algorithm shows an ability to define community structures even at low intrablock connectivity(prr = 0.02), although, like the other algorithms, it faces difficulties in clearly defining communities. However, as prr increases to 0.16 and 0.28, Leiden performs better at creating denser and more compact clusters. At full connectivity(prr = 1.00), has still many errors in the identification of the elements of each communities compares to the other algorithms performance.



(a) $prr = 0.02$ (b) $prr = 0.16$

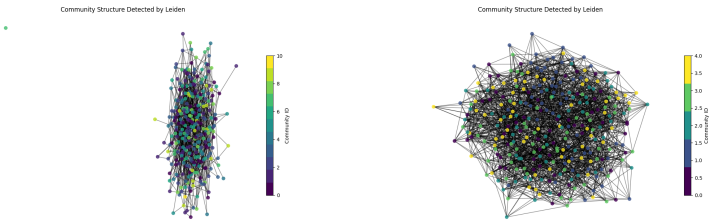
(c) $prr = 0.28$ (d) $prr = 1.00$

Figure 3.1: Agglomerative Greedy Algorithm Community Detection

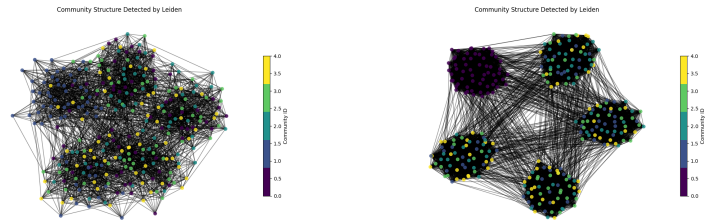
The Louvain algorithm, as shown in figure 3.3, which also shows imprecise community structures at a low prr of 0.02, improves markedly as prr increases. At prr = 1.00, Louvain does indeed identify distinct communities having all nodes of the same color in each cluster.

Similarly, the Infomap algorithm's, which visualization of results is shown in 3.4, performance improves with higher prr values. Although it starts with dispersed communities at prr = 0.02, by prr = 1.00, Infomap delineates sharply defined clusters. The algorithm's effectiveness becomes particularly evident with an increase in prr, reflecting its very good performance in extracting community structure in networks with significant intra-block connections.

The use of modularity (Q) to assess community structure in networks, as demonstrated by the results in the table 3.1, can help to understand the structure of a network under certain constraints. The modularity values reported by the Infomap, Louvain, Leiden and Agglomerative algorithms show that, while higher modularity usually correlates with clearer community structures, a modularity value such as Q=0.4 does not definitively confirm the presence of such a structure. For example, although Leiden's algorithm shows consistently low modularity at all 'prr' levels, which could imply a lack of community structure, this could also be indicative of the method's inability to optimise

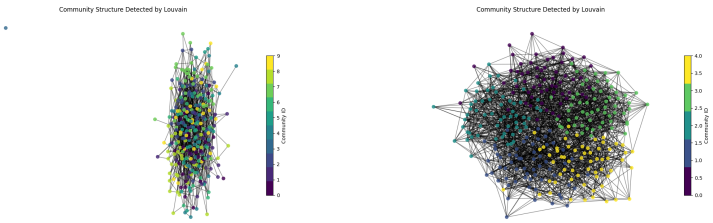


(a) $prr = 0.02$ (b) $prr = 0.16$

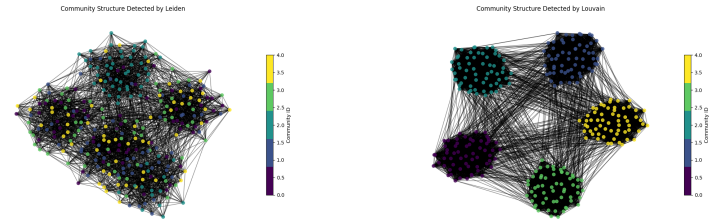


(c) $prr = 0.28$ (d) $prr = 1.00$

Figure 3.2: Leiden Algorithm Community Detection

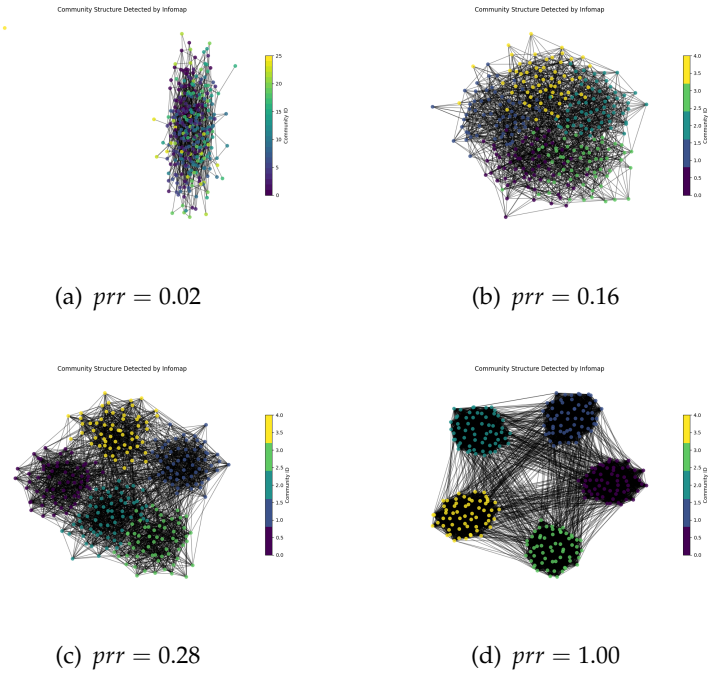


(a) $prr = 0.02$ (b) $prr = 0.16$



(c) $prr = 0.28$ (d) $prr = 1.00$

Figure 3.3: Louvain Algorithm Community Detection



(a) $prr = 0.02$ (b) $prr = 0.16$

(c) $prr = 0.28$ (d) $prr = 1.00$

Figure 3.4: Infomap Algorithm Community Detection

modularity effectively or to detect smaller communities.

Furthermore, increasing modularity values approaching Q=0.7 for Infomap, Leuven and Leiden with higher prr values suggest the existence of better defined communities. However, relying on modularity alone to assert community structure is risky due to the sensitivity of modularity to network size and the possible presence of subnetworks that do not necessarily constitute meaningful communities.

According to table 3.2, as the probability of intra-block connections(prr) increases, the algorithms show a trend towards perfect community detection, with Infomap and Louvain achieving a Normalized Mutual Information (NMI) and a Jaccard index of 1.0 with and a Normalized Variation of Information (NVI) at a prr of 0.16, indicating a perfect match with true community structures. This suggests that as communities become more internally connected, both Infomap and Louvain are very effective in detecting these communities.

Leiden's algorithm follows a similar trend but with worse results, as its performance measures increase steadily with prr. However, it does not achieve perfect scores. Leiden's performance is notably lower than that of Infomap and Louvain.

Table 3.1: Networks Numerical Descriptors

Network	Inf-mod	Inf-groups	Lou-mod	Lou-groups	Lei-mod	Lei-groups	Agg-mod	Agg-groups
0.00	0.422934	34	0.453362	11	0.016375	12	0.003921	288
0.02	0.000000	25	0.390164	11	0.000000	10	0.003776	297
0.04	0.000000	19	0.344778	11	0.000000	11	0.003729	299
0.06	0.000000	1	0.311171	11	0.011475	10	0.003663	300
0.08	0.000000	1	0.308135	9	0.011770	9	0.003612	300
0.10	0.000000	1	0.320817	6	0.013473	6	0.003594	300
0.12	0.376161	7	0.377652	5	0.027163	5	0.003568	300
0.14	0.430683	5	0.430671	5	0.026317	5	0.003537	300
0.16	0.458769	5	0.458769	5	0.029108	5	0.003521	300
0.18	0.482988	5	0.482988	5	0.033461	5	0.003517	300
0.20	0.503818	5	0.503818	5	0.059547	5	0.003491	300
0.22	0.525872	5	0.525872	5	0.061567	5	0.003475	300
0.24	0.541704	5	0.541704	5	0.067285	5	0.003469	300
0.26	0.557568	5	0.557568	5	0.074952	5	0.003458	300
0.28	0.571349	5	0.571349	5	0.069937	5	0.003449	300
0.30	0.583699	5	0.583699	5	0.070628	5	0.003446	300
0.32	0.593606	5	0.593606	5	0.072297	5	0.003442	300
0.34	0.604640	5	0.604640	5	0.075213	5	0.003432	300
0.36	0.613111	5	0.613111	5	0.070379	5	0.003424	300
0.38	0.620861	5	0.620861	5	0.084150	5	0.003418	300
0.40	0.628415	5	0.628415	5	0.089670	5	0.003409	300
0.42	0.635444	5	0.635444	5	0.092702	5	0.003401	300
0.44	0.641984	5	0.641984	5	0.101490	5	0.003397	300
0.46	0.647697	5	0.647697	5	0.111470	5	0.003392	300
0.48	0.653674	5	0.653674	5	0.114778	5	0.003388	300
0.50	0.658578	5	0.658578	5	0.116315	5	0.003383	300
0.52	0.663283	5	0.663283	5	0.116194	5	0.003382	300
0.54	0.667900	5	0.667900	5	0.122285	5	0.003378	300
0.56	0.671651	5	0.671651	5	0.122825	5	0.003377	300
0.58	0.675300	5	0.675300	5	0.123465	5	0.003376	300
0.60	0.678975	5	0.678975	5	0.124795	5	0.003372	300
0.62	0.682157	5	0.682157	5	0.124206	5	0.003371	300
0.64	0.685590	5	0.685590	5	0.126463	5	0.003368	300
0.66	0.688569	5	0.688569	5	0.125331	5	0.003364	300
0.68	0.691503	5	0.691503	5	0.129040	5	0.003362	300
0.70	0.694019	5	0.694019	5	0.130057	5	0.003359	300
0.72	0.696712	5	0.696712	5	0.142362	5	0.003357	300
0.74	0.699210	5	0.699210	5	0.147124	5	0.003356	300
0.76	0.701641	5	0.701641	5	0.148876	5	0.003353	300
0.78	0.703838	5	0.703838	5	0.147402	5	0.003351	300
0.80	0.705933	5	0.705933	5	0.147261	5	0.003351	300
0.82	0.708121	5	0.708121	5	0.148442	5	0.003349	300
0.84	0.710136	5	0.710136	5	0.149578	5	0.003347	300
0.86	0.711825	5	0.711825	5	0.149511	5	0.003346	300
0.88	0.713639	5	0.713639	5	0.148323	5	0.003344	300
0.90	0.715459	5	0.715459	5	0.150543	5	0.003343	300
0.92	0.717203	5	0.717203	5	0.150864	5	0.003342	300
0.94	0.718690	5	0.718690	5	0.150769	5	0.003340	300
0.96	0.720208	5	0.720208	5	0.152146	5	0.003339	300
0.98	0.721773	5	0.721773	5	0.159511	5	0.003338	300
1.00	0.723991	5	0.723991	5	0.176696	5	0.003336	300

The Agglomerative Greedy algorithm has an anomaly in its performance metrics. Despite varying prr levels, its NMI and NVI scores remain constant and do not reflect the perfect detection observed in other algorithms. Agglomerative Greedy method seems to be not as sensitive to the prr parameter as the other algorithms, possibly due to differences in the underlying methodology or implementation.

Table 3.2: Networks Numerical Descriptors

Network	Inf-nmi	Inf-nvi	Inf-jaccard	Lou-nmi	Lou-nvi	Lou-jaccard	Lei-nmi	Lei-nvi	Lei-jaccard	Agg-nmi	Agg-nvi	Agg-jaccard
0.00	0.068193	0.931807	0.061899	0.022848	0.977152	0.105149	0.062937	0.937063	0.140289	0.430872	0.569128	0.031894
0.02	0.091163	0.908837	0.100326	0.029371	0.970629	0.131980	0.069607	0.930393	0.137906	0.436192	0.563808	0.016667
0.04	0.110123	0.889877	0.125424	0.088928	0.911072	0.163671	0.048819	0.951181	0.145212	0.439159	0.560841	0.016667
0.06	0.000000	1.000000	0.200000	0.067482	0.932518	0.150606	0.052409	0.947591	0.142701	0.440145	0.559855	0.016667
0.08	0.000000	1.000000	0.200000	0.188818	0.811182	0.328070	0.062331	0.937669	0.162130	0.440145	0.559855	0.016667
0.10	0.000000	1.000000	0.200000	0.476740	0.523260	0.592867	0.096776	0.903224	0.197424	0.440145	0.559855	0.016667
0.12	0.771780	0.228220	0.792973	0.653118	0.346882	0.740095	0.119875	0.880125	0.215602	0.440145	0.559855	0.016667
0.14	0.971252	0.028748	0.980377	0.950220	0.049780	0.967529	0.130656	0.869344	0.224240	0.440145	0.559855	0.016667
0.16	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.139349	0.860651	0.233950	0.440145	0.559855	0.016667
0.18	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.150383	0.849617	0.241035	0.440145	0.559855	0.016667
0.20	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.158560	0.841440	0.245019	0.440145	0.559855	0.016667
0.22	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.166543	0.833457	0.248385	0.440145	0.559855	0.016667
0.24	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.175538	0.824462	0.256697	0.440145	0.559855	0.016667
0.26	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.184106	0.815894	0.256697	0.440145	0.559855	0.016667
0.28	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.184106	0.815894	0.256697	0.440145	0.559855	0.016667
0.30	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.184106	0.815894	0.256697	0.440145	0.559855	0.016667
0.32	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.184106	0.815894	0.256697	0.440145	0.559855	0.016667
0.34	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.184106	0.815894	0.256697	0.440145	0.559855	0.016667
0.36	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.184106	0.815894	0.256697	0.440145	0.559855	0.016667
0.38	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.189169	0.810831	0.263262	0.440145	0.559855	0.016667
0.40	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.201222	0.798778	0.274691	0.440145	0.559855	0.016667
0.42	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.201222	0.798778	0.274691	0.440145	0.559855	0.016667
0.44	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.216239	0.783761	0.288879	0.440145	0.559855	0.016667
0.46	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.232500	0.767500	0.298963	0.440145	0.559855	0.016667
0.48	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.238606	0.761394	0.304231	0.440145	0.559855	0.016667
0.50	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.238606	0.761394	0.304231	0.440145	0.559855	0.016667
0.52	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.239674	0.760326	0.308989	0.440145	0.559855	0.016667
0.54	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.239674	0.760326	0.308989	0.440145	0.559855	0.016667
0.56	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.239674	0.760326	0.308989	0.440145	0.559855	0.016667
0.58	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.239674	0.760326	0.308989	0.440145	0.559855	0.016667
0.60	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.239674	0.760326	0.308989	0.440145	0.559855	0.016667
0.62	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.239674	0.760326	0.308989	0.440145	0.559855	0.016667
0.64	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.239674	0.760326	0.308989	0.440145	0.559855	0.016667
0.66	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.239674	0.760326	0.308989	0.440145	0.559855	0.016667
0.68	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.239674	0.760326	0.308989	0.440145	0.559855	0.016667
0.70	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.239674	0.760326	0.308989	0.440145	0.559855	0.016667
0.72	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.255939	0.744061	0.322341	0.440145	0.559855	0.016667
0.74	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.265404	0.734596	0.328110	0.440145	0.559855	0.016667
0.76	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.265404	0.734596	0.328110	0.440145	0.559855	0.016667
0.78	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.265404	0.734596	0.328110	0.440145	0.559855	0.016667
0.80	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.265404	0.734596	0.328110	0.440145	0.559855	0.016667
0.82	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.265404	0.734596	0.328110	0.440145	0.559855	0.016667
0.84	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.265404	0.734596	0.328110	0.440145	0.559855	0.016667
0.86	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.265404	0.734596	0.328110	0.440145	0.559855	0.016667
0.88	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.265404	0.734596	0.328110	0.440145	0.559855	0.016667
0.90	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.265404	0.734596	0.328110	0.440145	0.559855	0.016667
0.92	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.265404	0.734596	0.328110	0.440145	0.559855	0.016667
0.94	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.265404	0.734596	0.328110	0.440145	0.559855	0.016667
0.96	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.265404	0.734596	0.328110	0.440145	0.559855	0.016667
0.98	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.276523	0.723477	0.336439	0.440145	0.559855	0.016667
1.00	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.320088	0.679912	0.352966	0.440145	0.559855	0.016667

3 | Characterization of the community structure of real networks

3.1 | Task Description

This second practical task focuses on the analysis of the community structure of a real network that captures interactions within a primary school. This network, provide information in both weighted and unweighted. Provides information on the dynamics of social interactions between students and

its peers, and teachers over a two-day period. In addition, node metadata specifying the school group to which each individual belongs allows for the examination of community structures in the context of existing social divisions within the school.

To address this task, we have used the Louvain algorithm on modularity maximization. This allows for the discovery of community structures within the network. By applying these algorithms to weighted and unweighted versions of the network, the impact of interaction intensity on the results of community detection can be explored. Furthermore, with the help of node metadata, it is possible to validate the composition of the detected communities in terms of the school groups they encompass. With this analysis, we want to understand the role of network weights in community detection and to understand how community structures manifest themselves in real-world social network dynamics.

3.2 | Implementation

To be able to develop this task, there were some efforts made,in order to keep consistent, on the programming language Julia. However, there were multiple challenges to overcome, like specific modules/packages without maintenance, some differences on the implementation of the algorithms, even limitations on handling the weighted version of the network, specifically on the pajek format.

For this reason, and in order to spend more time on the analysis and development of this report, rather than using it to fix some issues on the modules, we decided to develop the implementation in the Python language with NetworkX.

One thing to consider was that when reviewing the actual paper of the work, we realized that there are actually 11 different communities. Two for each of the five academic years of study, for example, 2A and 2B, and one of the teachers. However, since the Louvain method implemented, does not know a priori the number of communities, we decided to work with any given number.

Since early on in the process we realized that the number of communities that maximize the modularity was six, we decided to work with two different views/comparisons: the first one with all of the groups detailed on the txt file with their real community, and the second one grouping the 2 different groups for each academic year, this is the 2A and 2B groups, considered as a single group 2. In this way, we will be reducing the actual number of real communities from eleven to six. This is interesting, since the communities detected where also six. Our intuition told us that this could be the reason why the communities where less than the actual ones. This split was considered apart of the weighted and unweighted networks, therefore, at the end we ended up with four different cases:

- Unweighted with all eleven groups
- Unweighted with groups merged and reduced to six
- Weighted with all eleven groups
- Weighted with groups merged and reduced to six

Across the next section, and for convenience we will analyze them separately nd referring to this scenarios just as unweighted all, unweighted reduced, weighted all and weighted reduced. Also, all the code developed and the different tables, csv files and images in the folder follow this same nomenclature.

3.3 | Results

As it was previously mentioned, this analysis will be divided into 4 scenarios. For each of them, there are provided first a table, where the columns are the real groups the different people belong to while the columns are the predicted communities that the algorithm detected; second, a stacked bar plot, wherein a graphical way we can distinguish the composition of each of the real groups and more clearly the misclassifications; and finally, a brief yet concrete analysis where the main insights are discussed. The overall analysis will be provided as part of the conclusions.

3.3.1 | Unweighted with all eleven groups

Here it can be seen in table 3.3, that the inaccuracy is very clear, which is obvious since the algorithm only discovered 6 communities. We can see in each row, how the different groups are classified. And counterintuitive to what we thought, this does not occur specifically because of each academic year.

This can be seen more in detail when we analyze the figure 3.5, where we can see how the groups 1A and 2A were grouped in the same community, while also the groups 4A,5A, and 5B.

What is interesting, and makes total sense, is that the Teachers are classified in each of the colors. Since of course, there should be one teacher for each of the groups.

3.3.2 | Unweighted with groups merged and reduced to six

It is obvious that when we merge the groups since the algorithm is the same, we are just going to bring together the detailed view we saw in the previous section. But it is still very interesting, how it seems to be a trend that the first and second graders, are not that much social, or have too many interactions with the kids from the same academic year. This is possible since they are just acquiring some social skills, and this is more clear where we see the 5th year in figure 3.6.

Table 3.3: Unweighted network. Predicted vs all groups

Pred.	1A	1B	2A	2B	3A	3B	4A	4B	5A	5B	Teach
0	0	25	0	0	0	0	0	0	0	0	1
1	0	0	0	0	23	22	0	0	0	0	2
2	0	0	0	0	0	0	21	0	22	24	3
3	0	0	0	0	0	0	0	23	0	0	1
4	23	0	23	0	0	0	0	0	0	0	2
5	0	0	0	26	0	0	0	0	0	0	1

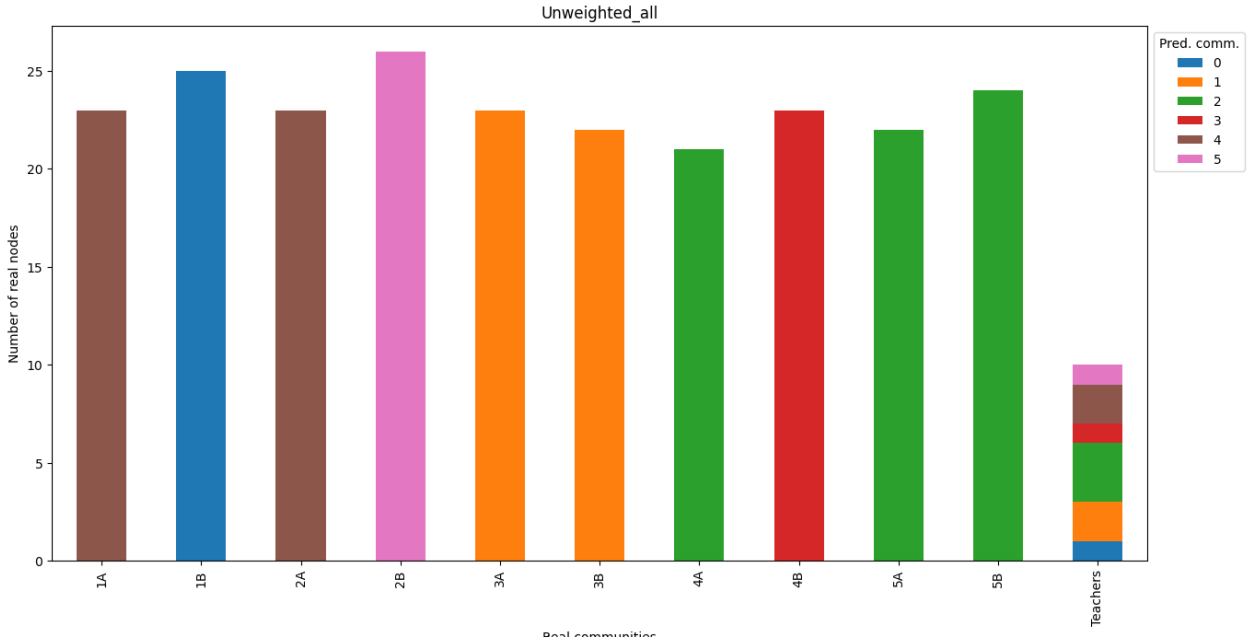


Figure 3.5: Unweighted network. True 11 groups vs 6 communities

3.3.3 | Weighted with all eleven groups

Here when we migrate to the weighted version of the network, it is obvious the huge impact that the weights of the connections have. Of course, it is not the same to have one interaction to have 20 interactions. And comparing the weighted tables and visualizations in the previous sections. It gets pretty clear. It can be seen in table 3.5, that the inaccuracy is reduced.

However, some specific behaviors now are clearer, like the one having the first graders. It seems that in a very recent incorporation to the school it is very common to only have contact with the children from their class. While when in advanced years, not only the interaction is limited to the class, but the academic year.

Table 3.4: Unweighted network. Predicted vs reduced groups

Pred.	**1**	**2**	**3**	**4**	**5**	**T**
0	25	0	0	0	0	1
1	0	0	45	0	0	2
2	0	0	0	21	46	3
3	0	0	0	23	0	1
4	23	23	0	0	0	2
5	0	26	0	0	0	1

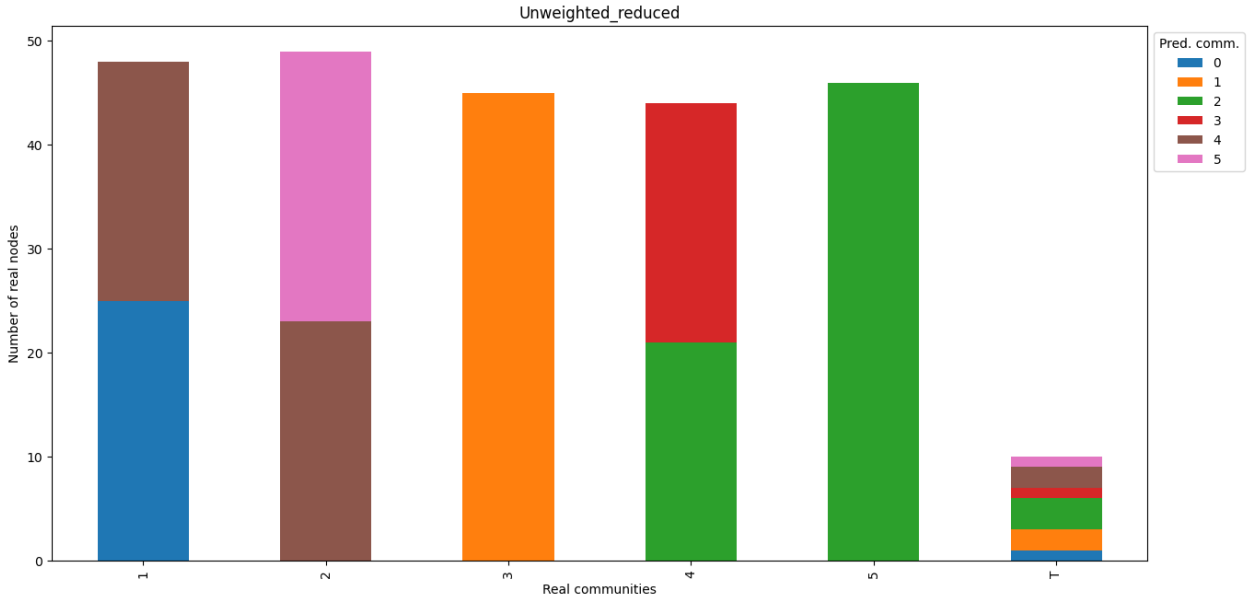


Figure 3.6: Unweighted network. Reduced 6 groups vs 6 communities

This can be seen more in detail when we analyze the figure 3.7.

Table 3.5: Weighted network. Predicted vs all groups

Pred.	1A	1B	2A	2B	3A	3B	4A	4B	5A	5B	Teach
0	0	0	0	0	0	0	21	21	0	0	2
1	0	0	0	0	23	22	0	0	0	0	2
2	0	0	0	0	0	0	0	2	22	24	2
3	0	25	0	0	0	0	0	0	0	0	1
4	23	0	0	0	0	0	0	0	0	0	1
5	0	0	23	26	0	0	0	0	0	0	2

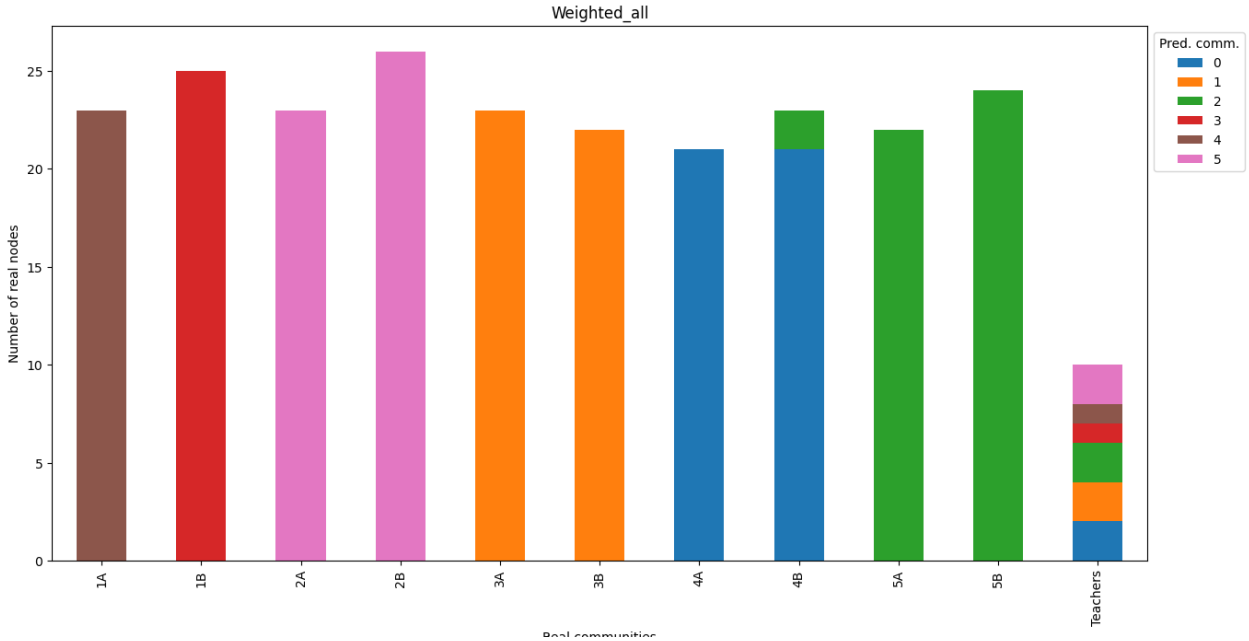


Figure 3.7: Weighted network. True 11 groups vs 6 communities

3.3.4 | Weighted with groups merged and reduced to six

Finally, and in order to complement the previous section, we see how this becomes clearer when we only have 6 groups. The 5 academic years, and the group of teachers.

We can see, in both, the table 3.6 and fig 3.8, that the groups from the first academic year were classified as different communities. That can be explained, as it was mentioned earlier, as a part of the development of social skills. If the French system is similar to the Spanish or Mexican system. These children are just 5 to 7 years old. It is obvious that it may be more difficult to interact with more children that are not even in the same classroom.

And this, with the fact that the teachers themselves are only ten people. Made more sense to have

each teacher incorporated with the class that is working, than with the other teachers.

However, there is a tendency in higher academic years, to have stronger interaction within the members of both groups. And these social skills seem to be even beyond the higher grades, where we see some of the children in the 4th year(4th column) in its majority are in color blue, be classified in the community of the children of the 5th year. This can be seen more in detail when we analyze the figure 3.8.

Table 3.6: Weighted network. Predicted vs reduced groups

Pred.	**1**	**2**	**3**	**4**	**5**	**T**
0	25	0	0	0	0	1
1	0	0	45	0	0	2
2	0	0	0	21	46	3
3	0	0	0	23	0	1
4	23	23	0	0	0	2
5	0	26	0	0	0	1

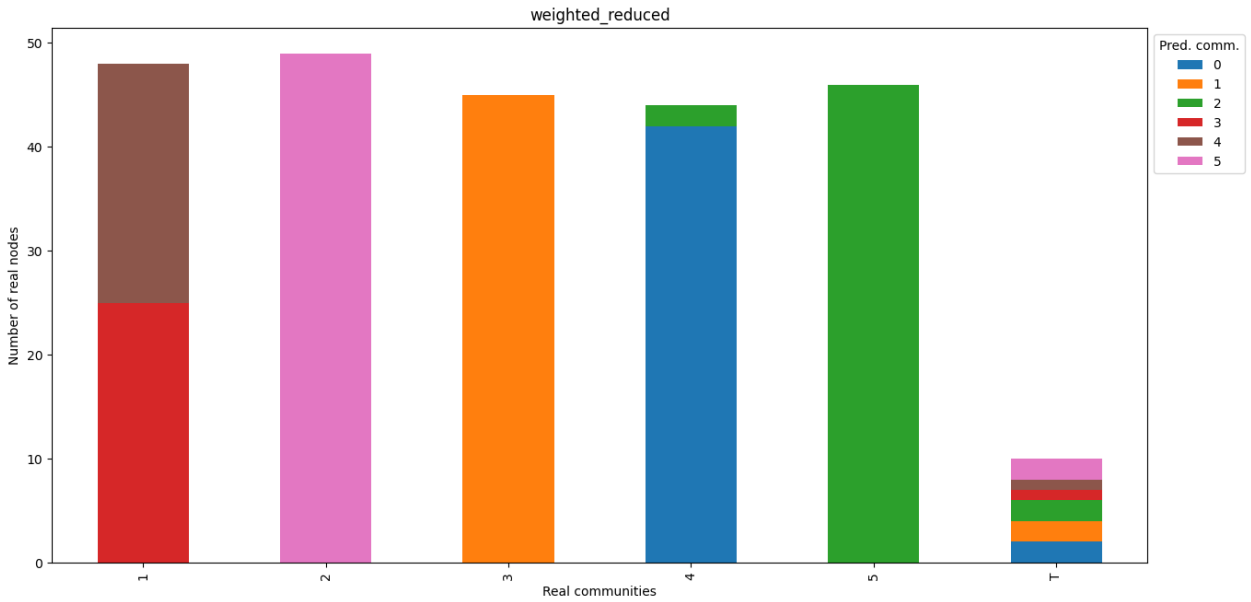


Figure 3.8: Weighted network. Reduced 6 groups vs 6 communities

4 | Conclusions

Community detection in networks plays a crucial role in understanding the underlying structure and dynamics of complex networks. By identifying communities or groups of nodes that are densely connected internally and sparsely connected externally, we can uncover patterns of interaction and organization within the network.

One significant importance of community detection lies in its ability to reveal hidden patterns and structures that might not be immediately apparent. For instance, in social networks, communities can represent groups of individuals with similar interests, behaviors, or roles. Understanding these communities can help in targeted marketing, recommendation systems, and even in predicting the spread of information or diseases.

Moreover, community detection aids in identifying key nodes or hubs within each community, which are crucial for maintaining cohesion and facilitating information flow. By focusing on these central nodes, we can devise strategies to enhance network resilience, optimize communication, or mitigate the impact of targeted attacks.

Howevwer, the main challenges is that there is no even a actual definition on what a community is. And of course, this has a huge impact on how it can be modeled mathematically. So one of the great challenges are, in fact, the definition of communities so then it can be discovered.

Through the assignment we had the opportunity to work with both synthetic and real networks. And it can be seen the complexities of their own.

For example, in the case of a real network, we could see how the proper design and recollection of the model of the network, like the edges, played a crucial role. It was evident the big change when we took them into account. But of course, there are some parts of the interactions between people that cannot be understood by simply analyzing the graphs. However, good insight can emerge from this even if there are some information missing.

In conclusion, community detection serves as a powerful tool for uncovering hidden structures and understanding the organization of complex networks. While it offers valuable insights, addressing the challenges of defining communities, handling dynamic networks, and ensuring scalability remains an ongoing area of research and development in network science.