Net-analysis-report

Alessio Pelizzoni

Contents

1 Introduction

The selected network is a Two-Mode (bipartite) graph, which means it consists of two distinct types of nodes, with edges only allowed between nodes of different types. In this case, the graph represents persons and crimes:

- **Left nodes**: persons who appeared in at least one crime case, either as a suspect, a victim, a witness, or even as both a suspect and a victim at the same time.
- **Right nodes**: crime events.

An edge between a person and a crime indicates that the person was involved in that crime. The edge also stores the role that the person had in that crime (victim, suspect, both, witness).

1.1 Projection

To simplify the analysis, I created a variant of the graph containing only one type of node. Specifically, I projected the bipartite graph onto the set of persons, resulting in a social network where two people are connected if they were involved in the same crime.

2 Analysis

In this chapter, all the measures computed on the graph are presented. Although the main focus of the report is on the projection of the original bipartite graph, the measures for both the bipartite graph and its projection are reported here.

2.1 Bipartite graph

For taking the measures, the graph has been analyzed both as single-type graph ('*' symbol) to confront it with the projection and a two-mode graph (using the `bipartite` module of Networkx).

Size:

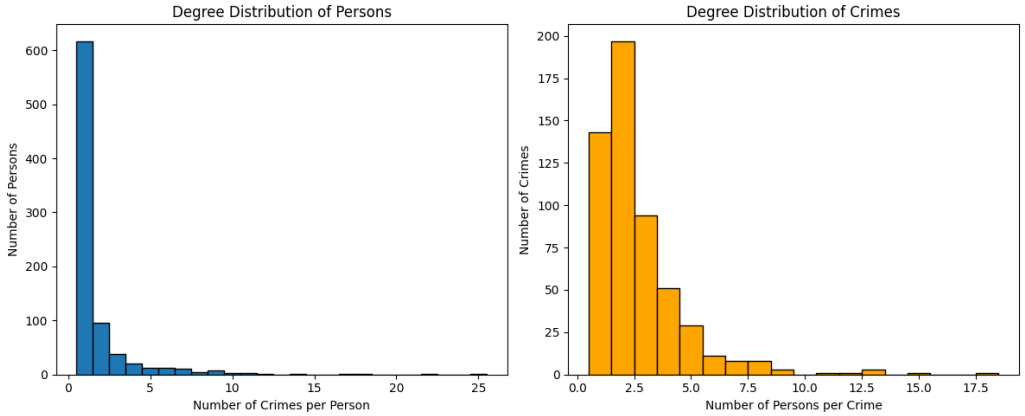
- **Nodes**: 1380, 829 left (person) and 551 right (crime)
- **Edges**: 1476

Measures:

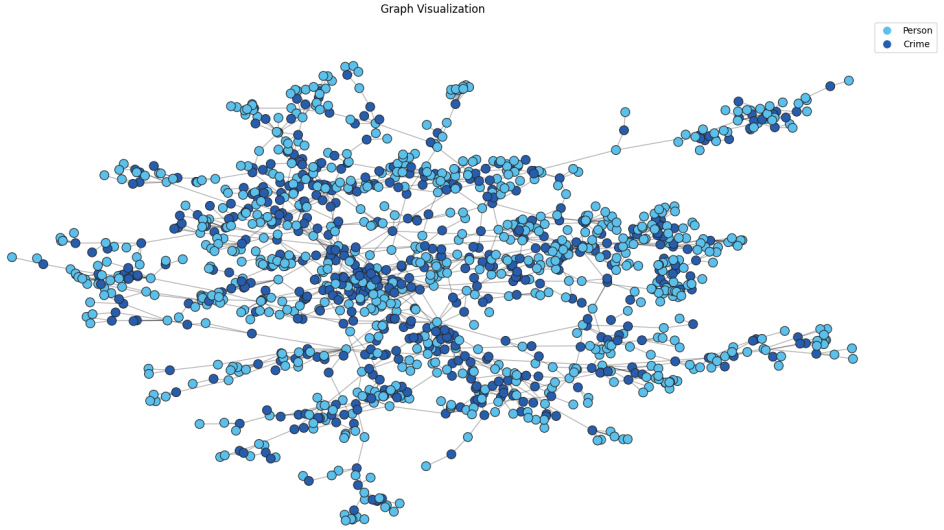
- **Bridges***: 1078
- **Local bridges***: 1476
- **Triangles***: 0
- **4-cycles**: 143
- **Average degree left (bipartite)**: 1.7804
- **Average degree right (bipartite)**: 2.6787
- **Average shortest path length**: 12.9561 (largest connected component)
- **Density (bipartite)**: 0.0032, both sides
- **Diameter***: 32 (largest connected component)
- **Average clustering (bipartite)**: 0.4275
- **Degree assortativity***: -0.1660

We can see that, even when using bipartite-specific functions, the graph remains very sparse. Interestingly, while the number of triangles is zero, the clustering coefficient is still moderate (0.42). This is because the bipartite clustering function does not count triangles (three nodes fully connected), but instead measures the density of 4-cycles (four nodes connected in a loop). This adaptation is necessary since, in bipartite graphs, each node's neighbors belong to the opposite partition, making triangles (formed by nodes of the same partition) impossible. It is also worth noting that the degree distribution deviates strongly from that of a random graph, showing instead a heavy-tailed shape closer to a power-law. This suggests the presence of hubs in the network: a few individuals or crimes are highly connected, while the

vast majority involve only a small number of links. This can mean that a small number of individuals are involved in many crimes, while most are involved in very few.



(a) Degree distributions



(b) Graph visualization

2.2 Projection graph

Size:

- **Nodes**: 819
- **Edges**: 2253

Measures:

- **Bridges**: 139
- **Local bridges**: 151
- **Connected components**: 10
- **Triangles**: 4312
- **Average triangles per node**: 15.7949
- **Average degree**: 5.5018
- **Average shortest path length**: 6.8175 (largest connected component)
- **Density**: 0.0067
- **Diameter**: 16 (largest connected component)
- **Average clustering**: 0.7279
- **Degree assortativity**: 0.1488

We can see that the number of person nodes has decreased from 829 to 819: this is because some individuals were only connected to isolated crimes and therefore disappeared in the process. The average degree and the average clustering coefficient both increase significantly. This reflects the fact that in the projection, people become more interconnected through shared crimes, generating a much denser local neighborhood.

Triangles, which were absent in the bipartite representation, now appear in large numbers (4312 in total, with an average of almost 16 per node). This exceeds the corresponding bipartite metric—the 4-cycle count—and is

a direct consequence of projecting co-occurrences: every neighborhood of a crime node can form a clique in the projection.

In terms of global structure, the diameter shrinks from 32 to 16, and the average shortest path length is reduced to 6.82, indicating that any two individuals are closer on average. Assortativity patterns also change. Degree assortativity becomes slightly positive (0.15), suggesting a slight tendency of individuals with similar degree to link together.

For the degree distribution, we have the same heavy-tailed behaviour we observed in the bipartite graph. This means that most people are involved with very few individuals in a crime, while a very small number is involved in many crimes (or a single/very few with a lot of victims/suspects/witnesses). There's also a clustering coefficient distribution graph, showing that most of the nodes are in a clique (high connectivity)

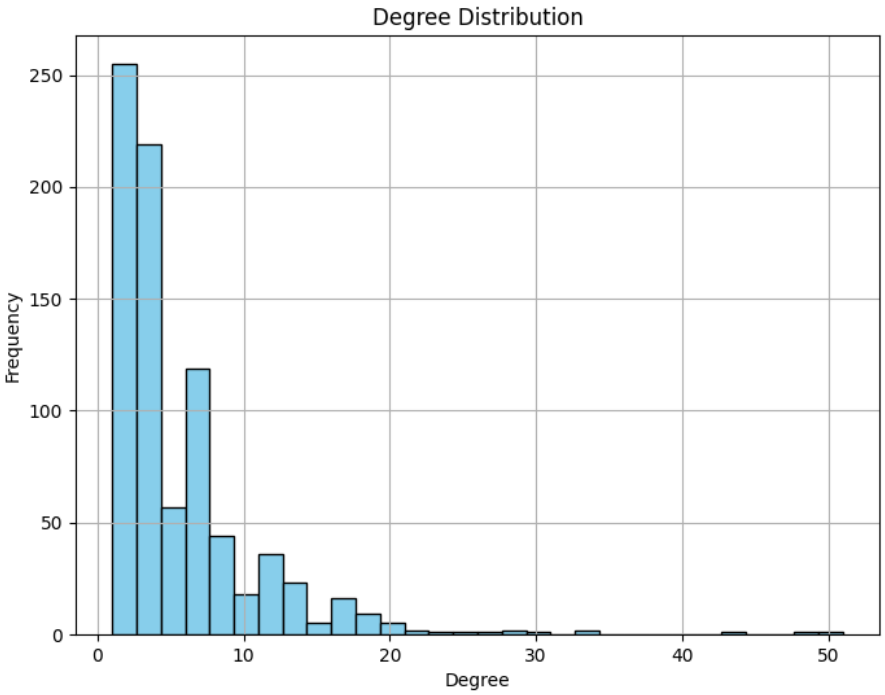


Figure 2: Degree distribution

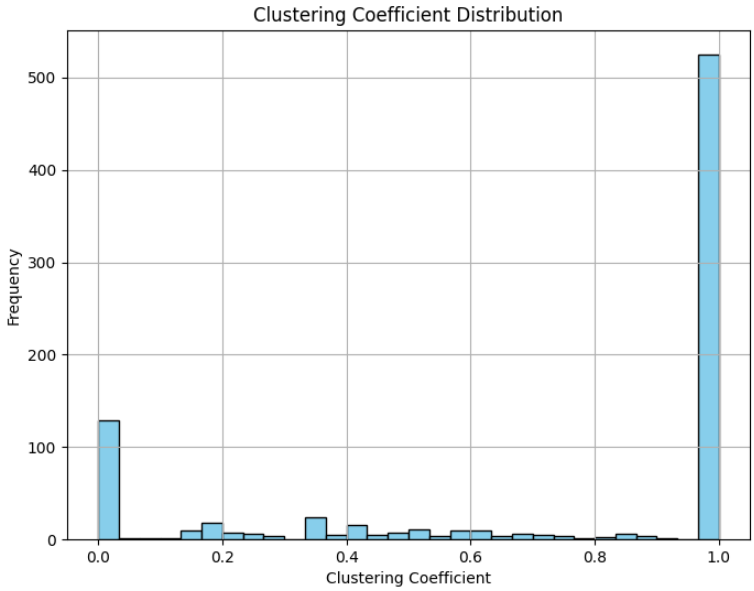


Figure 3: Clustering distribution

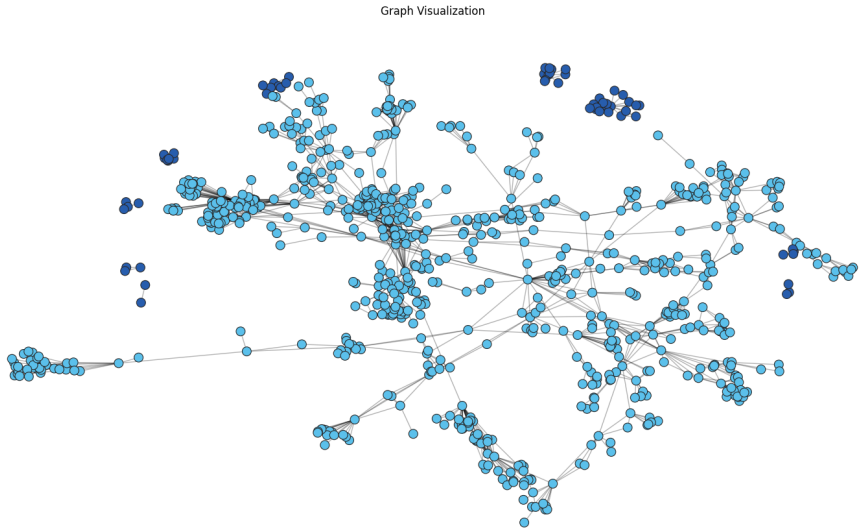


Figure 4: Graph visualization

3 Community detection

3.1 Detection

Here are described the methods used to detect communities, which will be then used to make a comparison of their result on the projected network. I measured the quality of the results my computing the modularity, the coverage and the performance of the resulting partitions.

- **Girvan-Newman**: divisive algorithm that detects communities by progressively removing edges from the original network. At each step, it removes the edge with the highest betweenness. Before applying it in the comparison, I first ran the algorithm while evaluating the modularity of the resulting partitions. Once the modularity dropped below zero, I selected the best partition obtained before that point, which occurred at the 26th step.
  - **Communities**: 37
  - **Modularity**: 0.8804
  - **Coverage**: 0.9596
  - **Performance**: 0.9565
  - **Execution time**: 124.5705 seconds
- **Louvain**: agglomerative algorithm for fast detection of communities. It optimizes network modularity without the need to do it ourselves.
  - **Communities**: 36
  - **Modularity**: 0.8825
  - **Coverage**: 0.9445
  - **Performance**: 0.9588
  - **Execution time**: 0.0746 seconds

- **Leiden**: another agglomerative algorithm, but resolves a problem of the Louvain algorithm: it guarantess that communities are well connected. It is also faster than Louvain.
  - **Communities**: 36
  - **Modularity**: 0.8869
  - **Coverage**: 0.9432
  - **Performance**: 0.9631
  - **Execution time**: 0.0228 seconds

From the results, we can see that all three algorithm produce a modularity value of 0.88, and coverage and performance values around 0.94-0.95. This means that all three results have good community structure (modularity), with most edges lying within a community, with only a few edges outside that link them (coverage), and that almost all node pairs are correctly classified (performance).

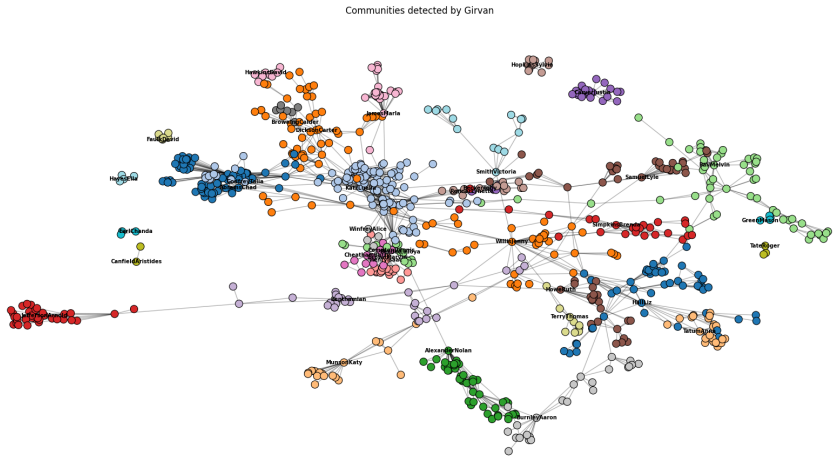


Figure 5: Communities found with Girvan-Newman algorithm

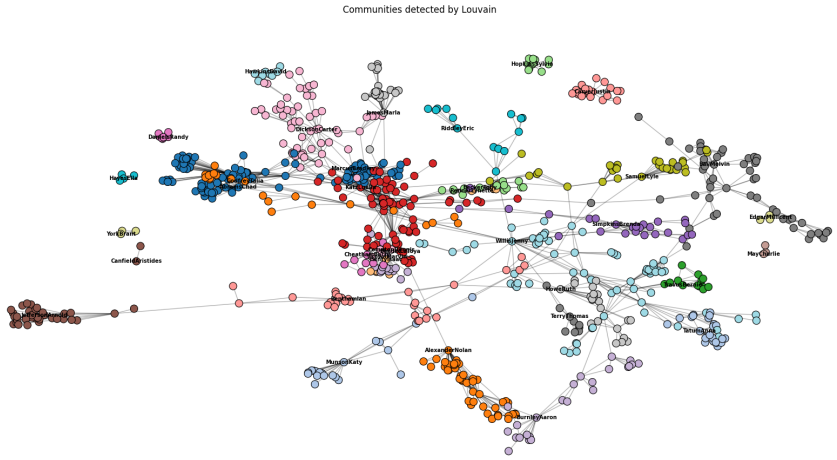


Figure 6: Communities found with Louvain algorithm

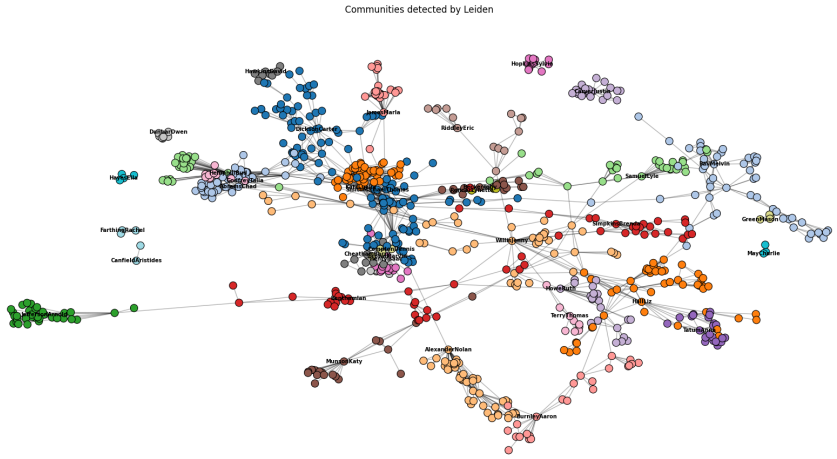


Figure 7: Communities found with Leiden algorithm

From these results, we can say that all relationship between people happen within small groups (coverage), implying that someone involvement in likely localized in one of these groups rather than spread across the whole network. The performance results indicate that these groups are likely meaningful.

Even without detailed background data, it is reasonable to assume that individuals within the same community may share some form of relationship, such as family ties, friendship, co-working, or geographical proximity.

The coverage also tells us, together with the clustering distribution, that probably there are some higly connected clusters inside the network, with few links between them.

3.2 Comparison

Here are the results of the comparison between the three algorithms. For this comparisons, I added two metrics: the Adjusted Rand Index (ARI) and Normalize Mutual Information (NMI), that measure the similarity between the two results.

```
--- Louvain Algorithm ---
Number of communities found: 35
Modularity: 0.8830
Coverage: 0.9445
Performance: 0.9590
Execution time: 0.0928 seconds

--- Girvan-Newman Algorithm ---

Partition at level 26 found with 37 communities
Modularity: 0.8804
Coverage: 0.9596
Performance: 0.9565
Execution time: 108.9596 seconds

ARI: 0.8896587790226422
NMI: 0.9692863123418851
```

Figure 8: Girvan-New compared with Louvain

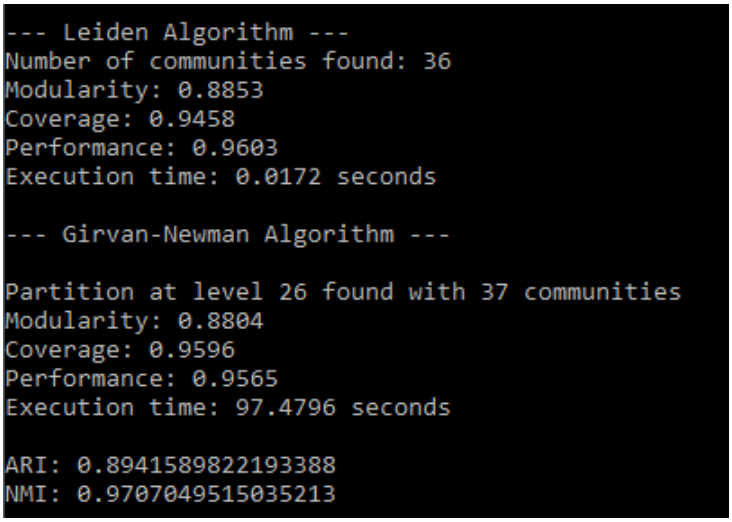


Figure 9: Girvan compared with Leiden

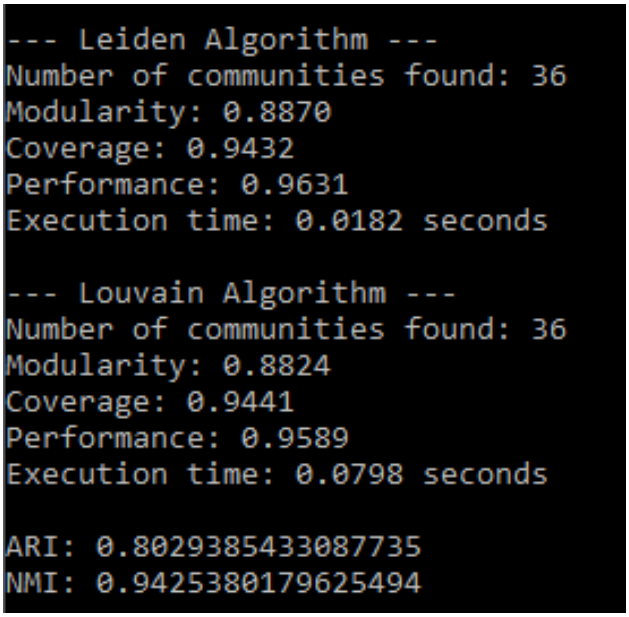


Figure 10: Louvain compared with Leiden

From the previous chapter and these results, we can see that the results are all of comparable quality, with only slight differences:

- **Community structure**: results produced by the Leiden algorithm show the greatest modularity (0.8853 – 0.8870), while the Girvan-Newman algorithm produces the lowest (0.8804);
- **Coverage/Performance**: Girvan-Newman has a higher coverage, suggesting its result captures more edges inside its communities, but it has the lowest performance, where leiden is the best.
- **Exectution time**: for this metric, the expectations were met: Girvan is by far the worst, and Leiden is faster than Louvain.
- **Similarity**: we can see that ARI is lower in all three cases, and this is probably because it considers that two partition could share some similarity by chance, and it adjust for that. Both Leiden and Louvain agree the most with Girvan, while both scores are lower between the two of them (Leiden and Louvain).

From these scores, we can see that the results show some consistency between these algorithms, adding value to the high modularity, coverage and performance observed before.

4 Message spreading

4.1 Introduction

In this chapter, the spreading process of messages through a network will be analyzed. At the start, there are initial nodes, called **gossipers**, which begin to spread the message: "This is my secret message". At each step, every node will accept the message if a fraction (threshold) of its neighbors has already received the message.

Before starting, a small percentage of all nodes (who are not gossipers) are chosen as malicious nodes: if they accept the message, they will tamper with it and forward it instead of the original. The tampering function changes for every malicious node: the phrase is modified by randomly choosing a word from the original string and replacing it with a random sequence of characters of the same length. With this method, the message can be tampered with multiple times. Examples are provided below:

- this aj my secret rfwhkzx;
- this is ps secret rfwhkzx;
- this is xb secret message;

I did four simulations:

- **Simulation 1, 3**: the ten nodes with the highest degree are considered the initial gossipers, using a threshold of 0.2. For the malicious nodes, I increased their fraction in the simulation until the number of nodes receiving the tampered message exceeded those receiving the original one.
- **Simulation 2, 4**: the same method was used, but in this case the initial gossipers were selected at random.

For each simulation I produced a heatmap that represent the cosine similarity betweenn all the messages known in the network. Here's an example with the Zachary's Karate Club network:

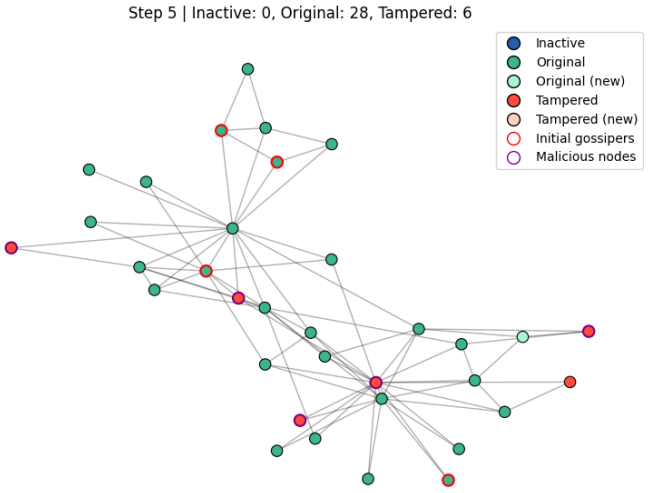


Figure 11: Final step of simulation

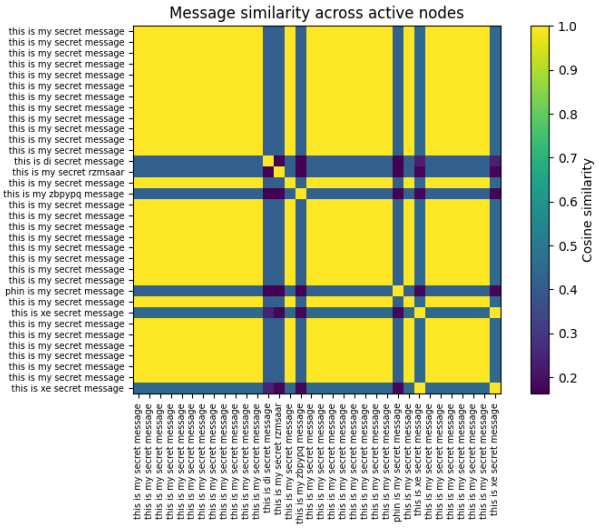


Figure 12: Messages heatmap

4.2 Simulation

- **Simulation 1**: top ten highest degree as gossipers, threshold value of 0.2, 10% malicious nodes
  - **Inactive**: 488
  - **Original**: 269
  - **Tampered**: 62

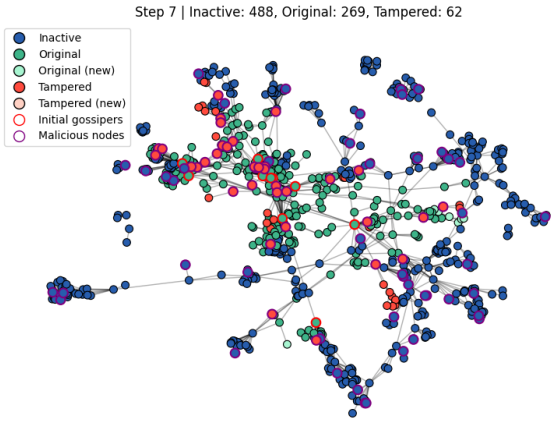


Figure 13: Animation

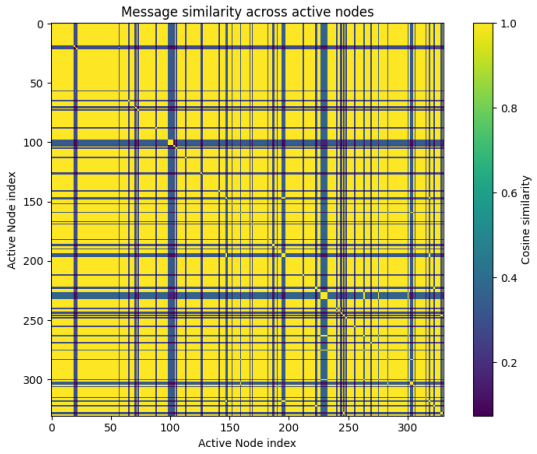


Figure 14: Messages heatmap

- **Simulation 2**: fifteen randomly chosen nodes as gossipers, threshold value of 0.1, 7% of malicious nodes
  - **Inactive**: 118
  - **Original**: 590
  - **Tampered**: 111

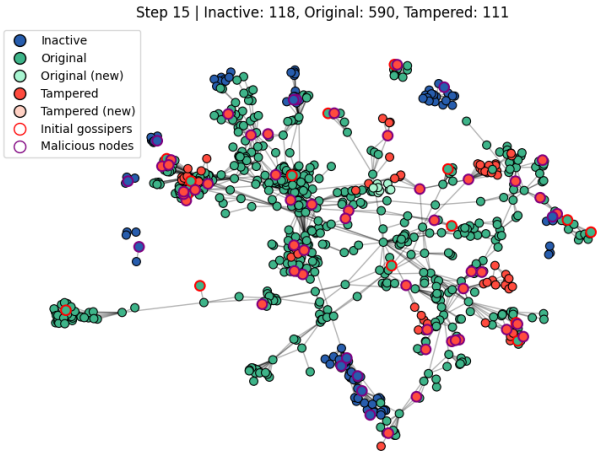


Figure 15: Animation

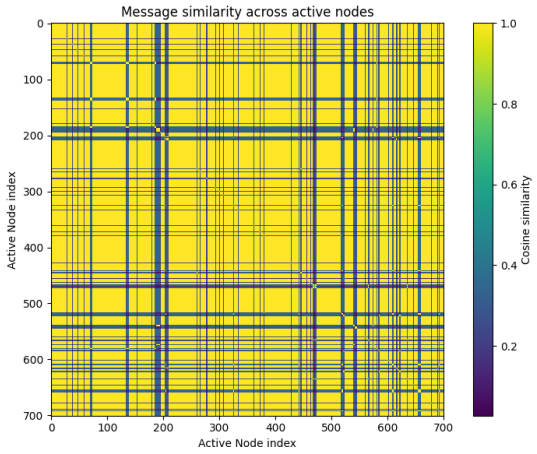


Figure 16: Messages heatmap

- **Simulation 3**: same gossipers and threshold as first simulation, but with 45% of malicious nodes.
  - **Inactive**: 503
  - **Original**: 131
  - **Tampered**: 185

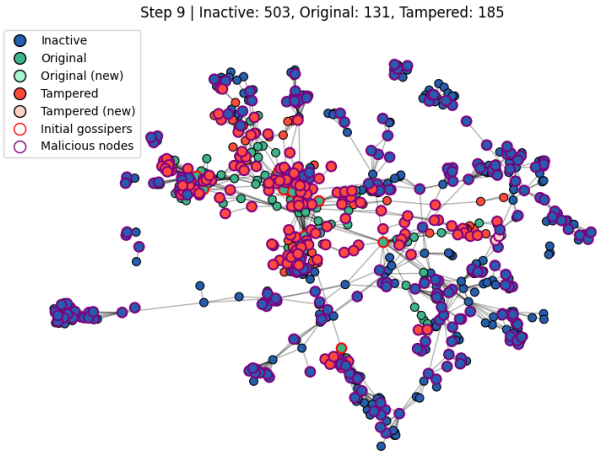


Figure 17: Animation

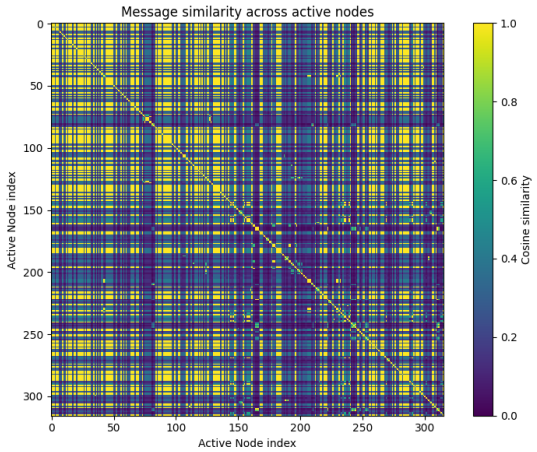


Figure 18: Messages heatmap

- **Simulation 4**: same gossipers and threshold as second simulation, but with 20% of malicious nodes.
  - **Inactive**: 129
  - **Original**: 318
  - **Tampered**: 372

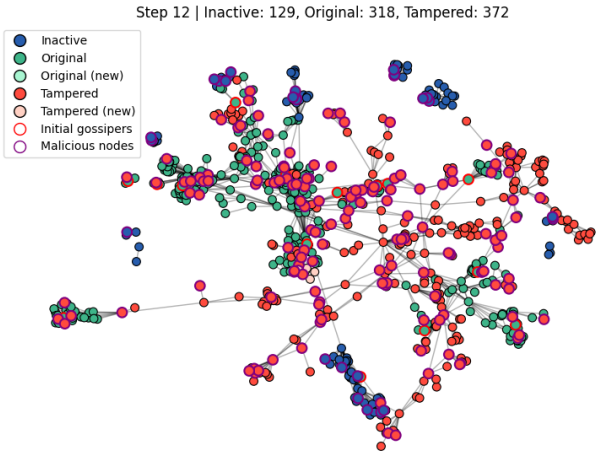


Figure 19: Animation

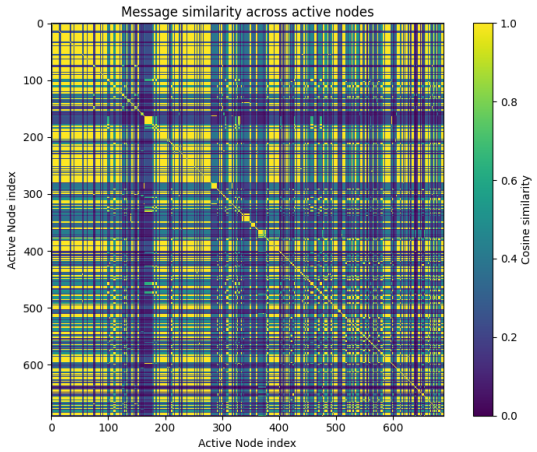


Figure 20: Messages heatmap

4.3 Conclusions

As we can see from the animations, the threshold to accept the message is always 0.2 (20%) or lower: this further confirms the idea that the network is likely made of highly connected clusters of nodes, because the message can spread within clusters but struggles to pass through sparsely connected areas. Even with a value of 0.1, the message doesn't reach all the nodes within the largest connected component. I tried other runs with even lower thresholds, and I found that with a value of 0.06 (6%), the largest component can be fully covered. Obviously, the other disconnected components cannot be reached unless there is an initial gossiper within them, which would likely require a high number of initial gossipers.

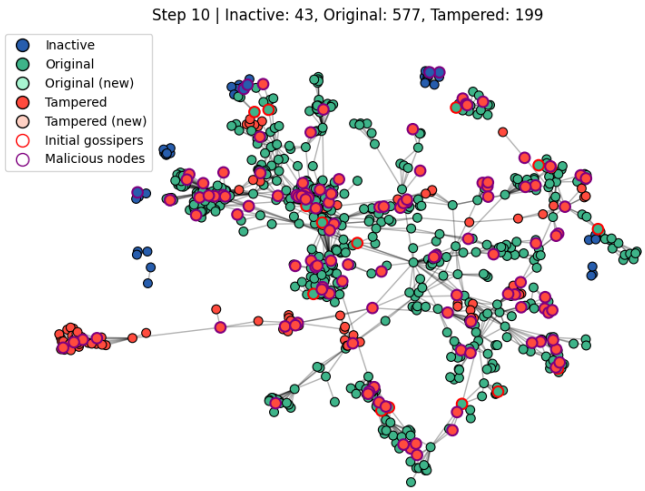


Figure 21: 15 random gossipers, threshold: 0.06, 15% malicious

We can also note that there is a significant difference between the two methods: while in the first the original message spreads much faster, in the second this is not always the case. This means that, to exceed the number of nodes that accept the original message, the fraction of malicious nodes must be sufficiently high to increase the chance of appearing in the right nodes. This is not true in the second method: here the original message usually does not expand at the same rate, and the malicious nodes have a higher chance of 'isolating' its spread and forwarding their tampered one.

5 Robustness

5.1 Introduction

Here, we examine the robustness of our network and test it against two types of attacks:

- **Random failures**: removal of nodes at random
- **Target attacks**: removal of targeted nodes, determined by a certain measure (degree, betweenness, closeness, pagerank, ...)

The network will be tested by running a simulation for each of these methods, and the size of the giant component will then be plotted with respect to the percentage of removed nodes. Below is the result of the simulation on the selected network:

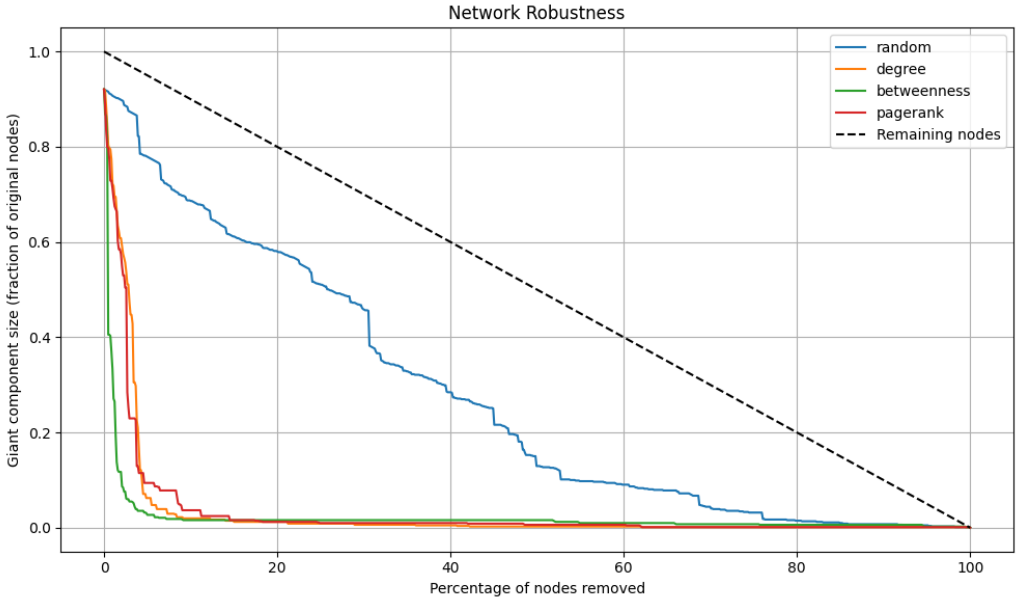


Figure 22: Simulation on the unchanged network

I also computed the critical threshold to measure the robustness against random failures and compared it with that of a random network having the same number of nodes and edges:

Name	Nodes	Edges	$< k >$	$< k^2 >$	f_c
Original	819	2253	5.5018	59.7509	0.8986
Randomized	819	2253	5.5018	35.9487	0.8193

From the table above, we observe enhanced robustness in the network, with both the critical threshold and the second moment exceeding those of a randomized network of the same size.

After running the simulation, the network will be modified using strategies aimed at increasing its robustness, and then tested again under the same conditions.

5.2 Building robustness

From the graph displayed in the previous chapter, we can observe that, while the network is somewhat resistant to random failures, this is not the case with targeted attacks: by removing a few nodes, we can see that the size of the giant component collapses immediately. This means that the giant compoment is extremely dependent on high degree, high betweenness nodes.

- **1. Form a clique around top-degree nodes**: link all the neighbors of the highest-degree nodes together to form a clique. With this method, even if the central node is removed, the neighbors remain connected. I applied this to the three nodes with the highest degree.
- **2. Form a clique around top-betweenness nodes**: same idea as the first strategy, but the centrality measure is betweenness, not degree. The objective is to 'split' the betweenness of these nodes among their neighbors.
- **3. Reinforce bridges**: a bridge is an edge that, if removed, increases the number of connected components. This strategy connects the neighbors of the bridge nodes. It increases the number of possible paths and the local clustering of nodes, reducing their betweenness.
- **4. Link communities**: in section 3.1, it was theorized that the network is composed of several loosely connected groups with a high density of internal edges. The strategy detects all communities (using

Leiden) and connects each pair of them with one edge. The objective is to increase bridges between communities and avoid reliance on one or a few nodes for external links.

Below are diplayed the simulation graphs and computed thresholds for all strategies.

Strategy 1:

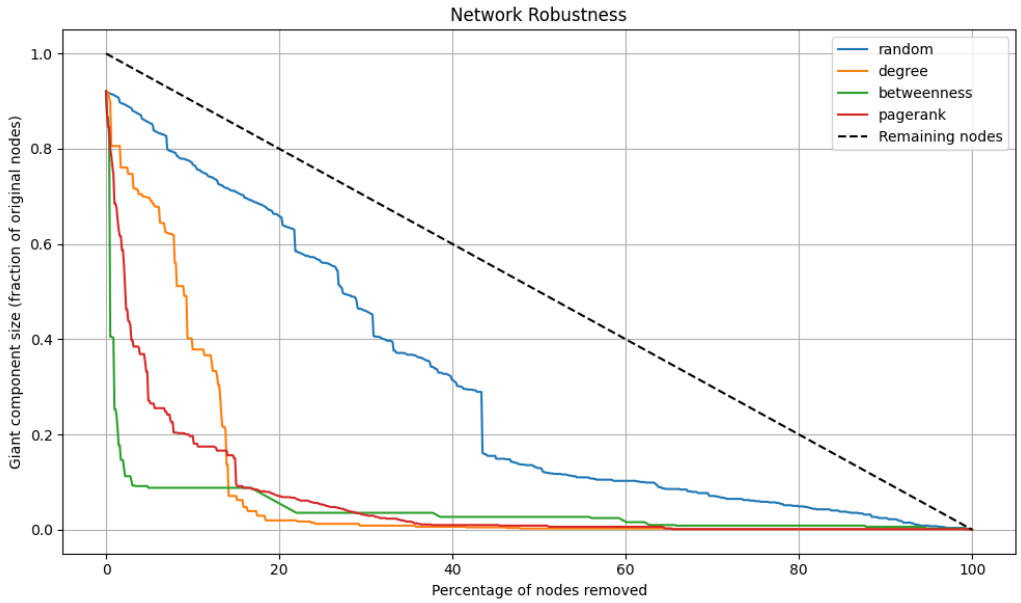


Figure 23: Simulation of strategy 1

Name	**Nodes**	**Edges**	$< k >$	$< k^2 >$	f_c
Original	819	2253	5.5018	59.7509	0.8986
1st clique	819	3234	7.8974	219.9756	0.9628
2nd clique	819	4232	10.3346	366.3687	0.9710
3rd clique	819	5964	14.5641	796.4200	0.9814

Strategy 2:

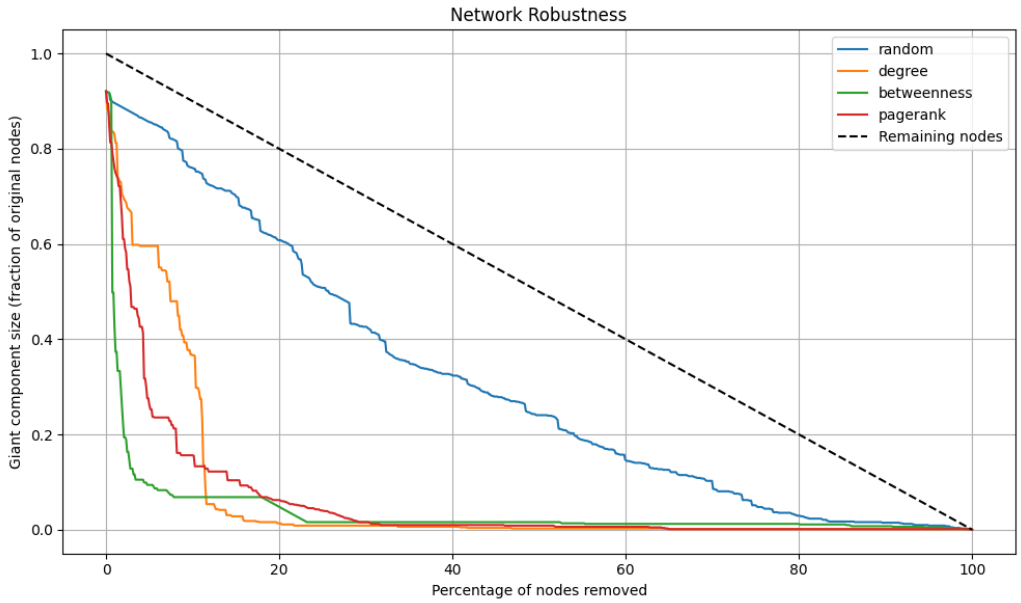


Figure 24: Simulation of strategy 2

Name	**Nodes**	**Edges**	$< k >$	$< k^2 >$	f_c
Original	819	2253	5.5018	59.7509	0.8986
1st clique	819	2586	6.3150	87.1600	0.9219
2nd clique	819	2823	6.8938	123.9683	0.9411
3rd clique	819	5186	12.6642	683.5849	0.9811

Strategy 3:

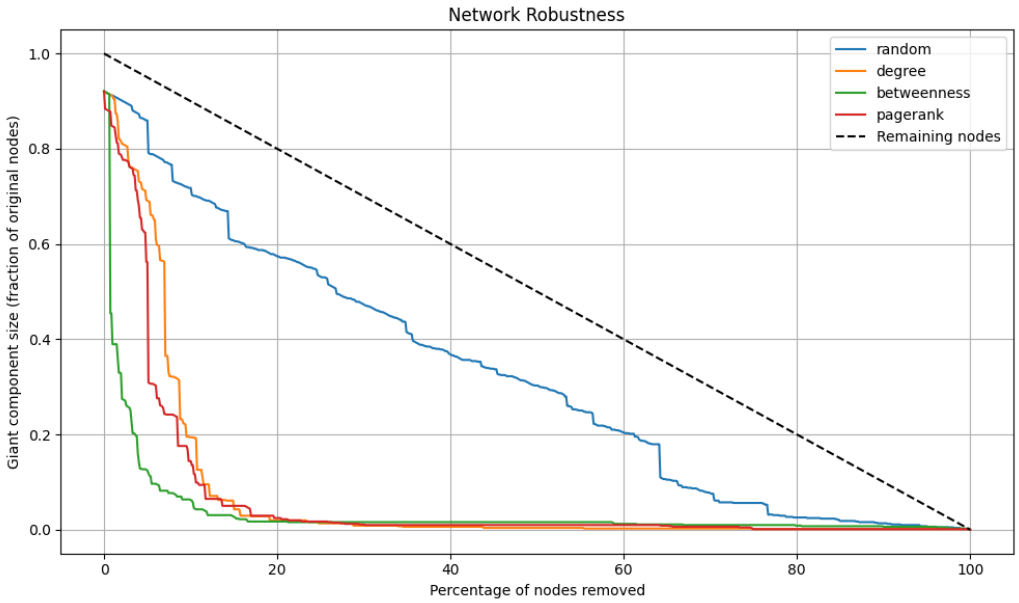


Figure 25: Simulation of strategy 3

Name	**Nodes**	**Edges**	$< k >$	$< k^2 >$	f_c
Original	819	2253	5.5018	59.7509	0.8986
Reinforced	819	4677	11.4212	258.2662	0.9537

Strategy 4:

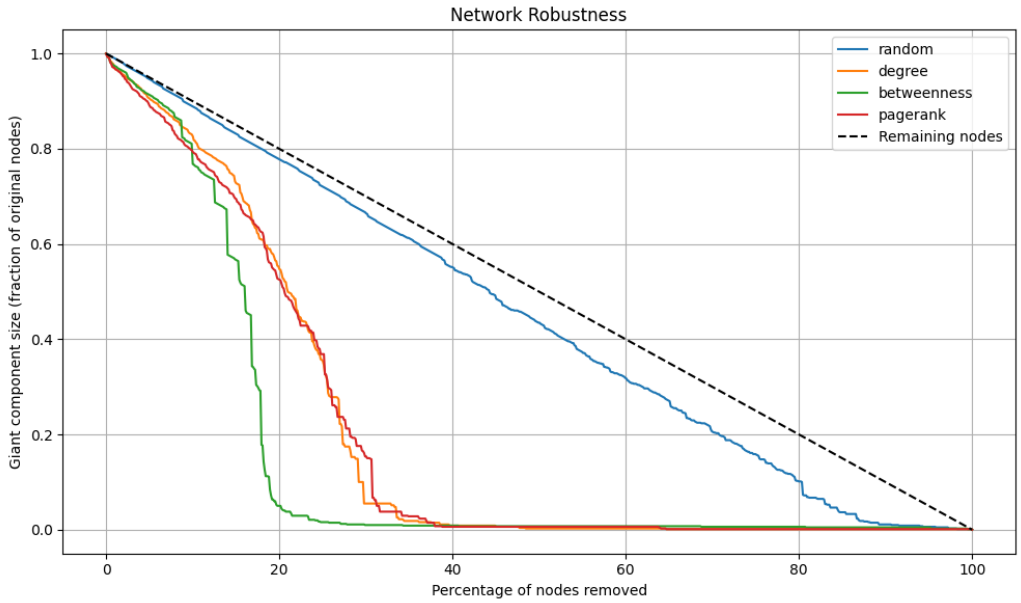


Figure 26: Simulation of strategy 4

Name	**Nodes**	**Edges**	$< k >$	$< k^2 >$	f_c
Original	819	2253	5.5018	59.7509	0.8986
Reinforced	819	2883	7.0403	81.5726	0.9055

5.3 Conclusions

From the graphs, we can see that the first three strategies produce slightly better results than the original graph. They also show a significant increase in the value of the critical threshold (up to 0.98). The fourth strategy is different: while its critical threshold does not increase as much as the other strategies (almost 0.1), the graph shows that the network is much more robust to targeted attacks. Another difference is the cost (added edges): while the

other strategies add between 2,400 and 3,700 edges, the fourth adds only 630 edges.

We can also observe that, in the reifonrced network, the degree of most nodes increases (average degree: 7.04), as can be seen by comparing the distribution of the original network in Figure 2 with the one below:

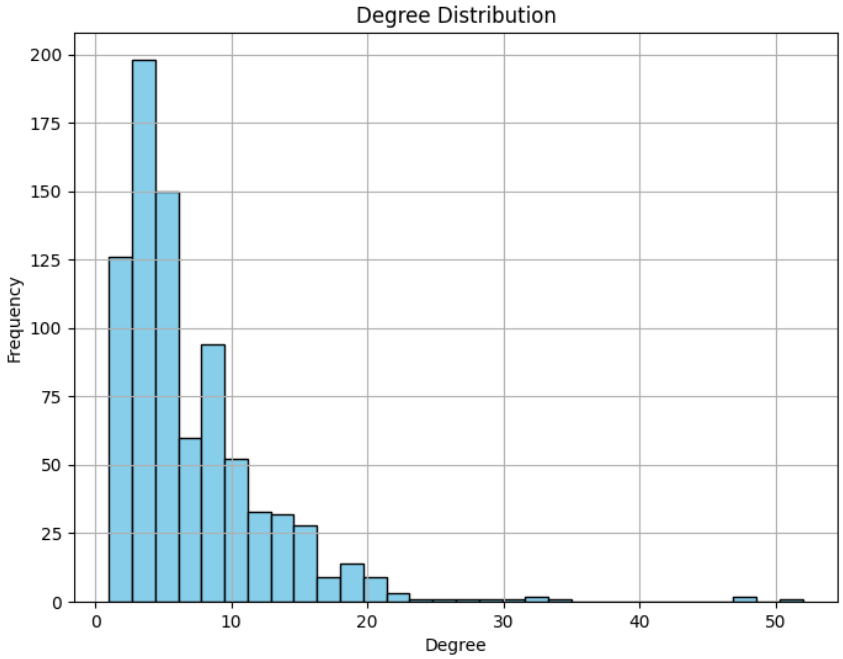


Figure 27: Degree distribution of the enhanced graph

This strategy shows better resistance to targeted attacks than the other strategies. This likely happens because the isolated communities do not rely on one or a few nodes for their external links (the betweenness drops from a maximum value of 0.35 to 0.06), and if an important node fails, the community is still connected to the rest of the network. This strategy treats communities as 'giant nodes' and creates a clique among them. This is also why the average clustering drops from 0.7 to 0.4: the densely connected, isolated clusters are no longer isolated. This probably lowers many nodes' local clustering coefficients. There's also another cost for this strategy: the structure of the graph changes significantly.