Ecole Polytechnique de Louvain

Homework 1 : Module "Networks"

Lallemand Martin , 25071800

Lemaire Louis, 37341800

Ndamtang Manuelle, 66732000

October 20, 2021

Contents

1 Introduction

In this report, we will analyse the links between the characters of the movie "Avengers : Endgame". We have chosen this movie because it contains a lot of characters with many interactions between them. We start by generating a graph containing all the characters as nodes and an edge is drawn if the characters appear in the same scene (at least once), using the movie screenplay.

In the following sections, we will by briefly explain how we managed to build this network. We will use the Louvain algorithm to find communities in the network, then apply the k-core decomposition to it. We will also compare the properties of our network and those of a Barabasi-Albert model of the same average degree and size.

We will then solve an influence maximisation problem using the independent cascade model with different starting sets : the one chosen by the greedy algorithm, the nodes with the highest degree and a random set, all three of the same size (5% of the nodes).

See the source code of the project:
`https://github.com/marty12342000/LNIMA2472-Homework-1/blob/main/GroupCode.ipynb`

2 Data management, network construction and results

The screenplay can be found as a .pdf file on the internet (`https://thescriptsavant.com/free-movie-scre` we then convert it to .txt with an online tool, as text files are much easier to read and manipulate with Python.

The next steps are identifying the scene boundaries, splitting the text in scenes using them, extracting all the characters appearing in each scene, then create a 58x58 dataframe containing the numbers of co-appearances between the characters.

See below the head of the interactions dataframe :

	STEVE	TONY	THOR	BRUCE BANNER	NATASHA	CLINT BARTON	RHODEY	CAROL DANVERS	SCOTT LANG	ROCKET	...	CULL OBSIDIAN	CORVUS GLAIVE	RED SKULL	BROCK RUMLOW
STEVE	0	29	19	3	19	7	18	12	15	16	...	1	1	0	2
TONY	0	0	20	1	11	8	11	6	13	12	...	1	1	0	1
THOR	0	0	0	3	9	6	15	5	9	20	...	1	0	0	1
BRUCE BANNER	0	0	0	0	4	0	3	3	0	3	...	0	0	0	0
NATASHA	0	0	0	0	0	11	12	9	11	10	...	0	0	2	1

Figure 1: Head of the characters interactions dataframe

Finally, we build an interaction network with the python package "networkx" using the aforesaid dataframe.

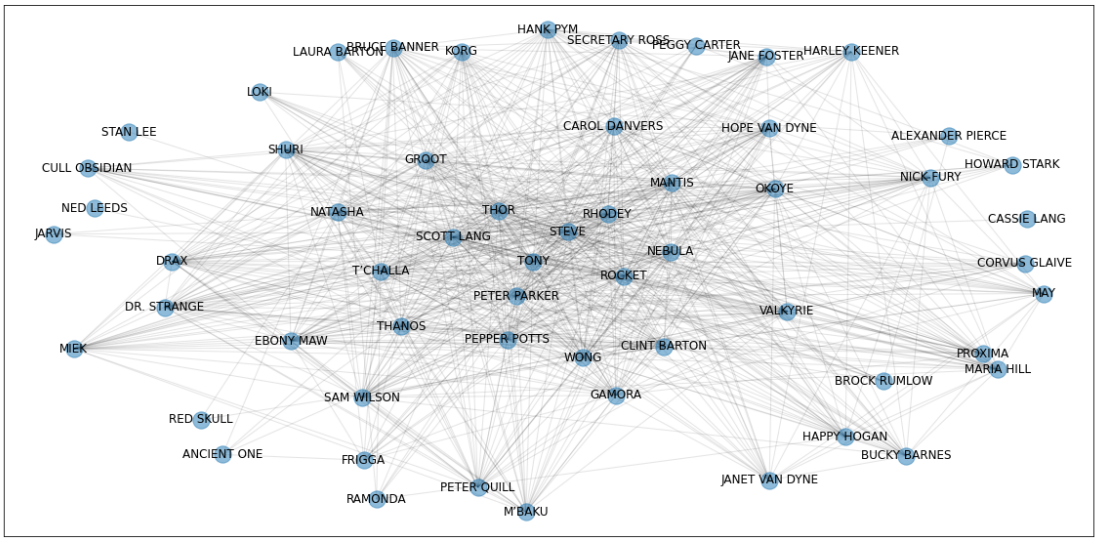

Figure 2: Interaction network

At first sight, it is obvious that the characters are very interconnected, considering the amount of edges that makes this graph hardly readable. In order to analyze this graph, we now need to use different algorithms to highlight some of its properties.

3 Analyzing the graph

3.1 The degree assortativity and the community detection

To find assortativity we use an existing function of the Networkx library, used to study our network. This function computes degree assortativity of graph and measures the similarity of connections in the graph with respect to the node degree.
The value obtained being -0.2, we tend to have a dissociative network, i.e. many actors with dissimilar degrees within the network. In other words, there are many actors with more influence interacting more compared to other actors with less influence. One could assume that this is the case of more famous and active superheroes compared to less influential superheroes and ordinary actors.

The detection of the communities by the use of the Louvain algorithm was done with the help of an already existing library named "python-louvain". The result obtained confirms more or less the conclusion held by the degree of assortativity. We notice 3 communities of actors that could be described as superheroes and personalities that are very popular and face each other regularly in the movie, another community of superheroes that are active but do not have huge influence (at least in the movie) and personalities that are there but do not interact much in the movie. We see that there isn't a lot of different communities and it's normal because this movie represent the end of a huge saga, so in this film all the characters interact with each other in order to conclude the long serie of movies.

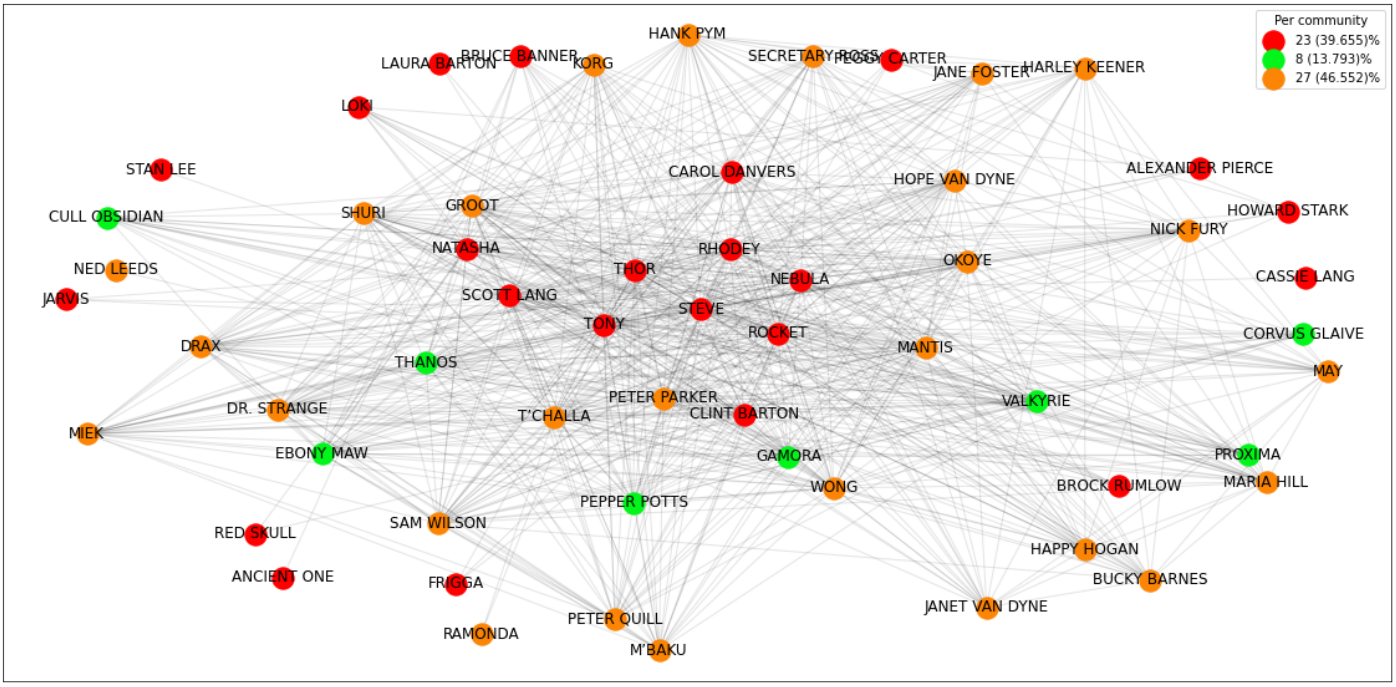

Figure 3: Interaction network with communities

3.2 K-core decomposition

The implementation of k-core decomposition was done from scratch in python. The particularity is that it is based on copying the graph and deleting as soon as a node is in a shell.

K-core decomposition reveals the hierarchies in the network and how central the nodes in the network are. The result is :

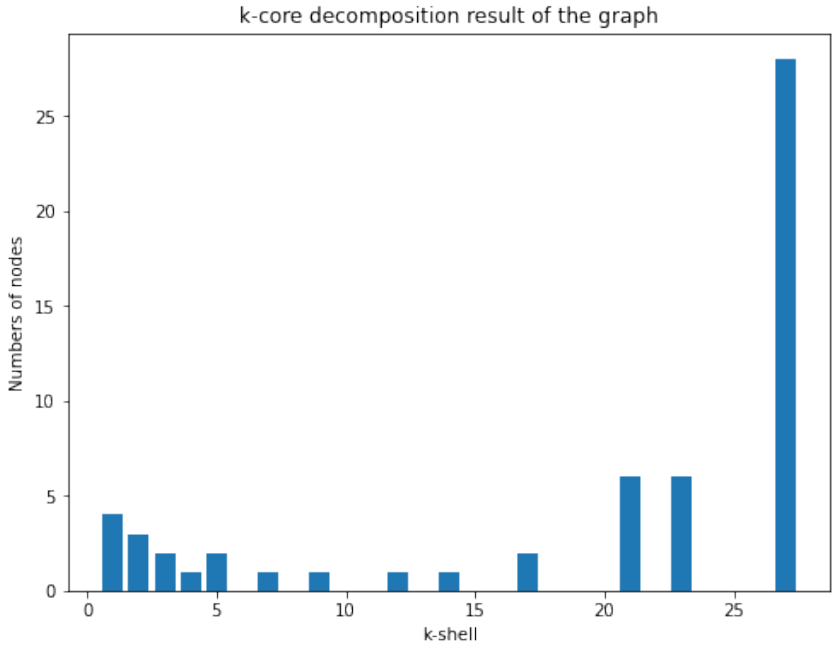

Figure 4: K-core decomposition

Here is a visualisation of each shell, the red nodes being in the outer k shells and the green nodes in

the inner k shell :

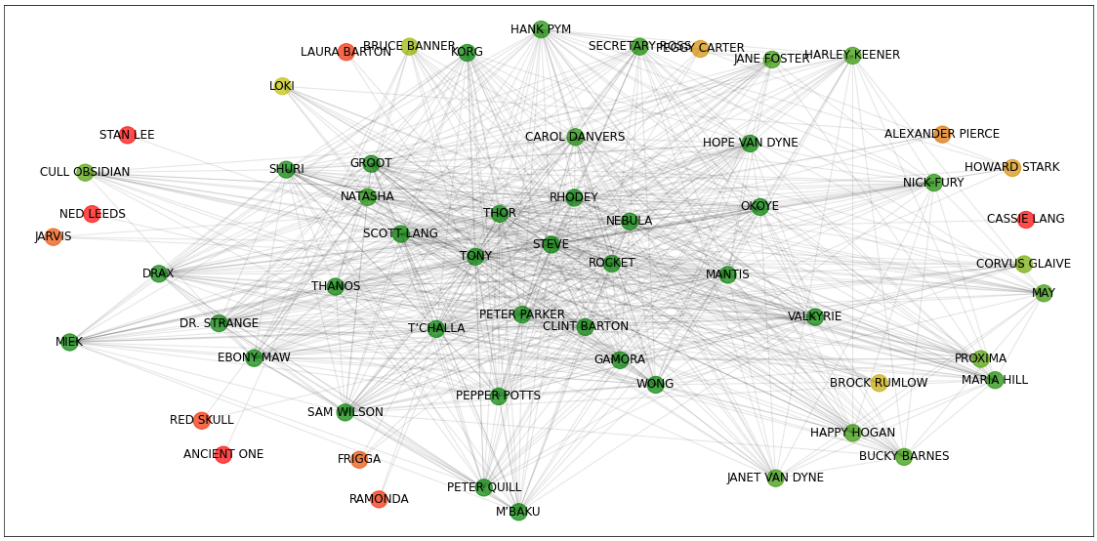

Figure 5: Illustration of the k shells

As you can see, there are many interconnected nodes, more than 15 to 27 nodes out of 58. This means that in our network we have a big cluster(many hubs) of actors interacting with each other on several scenes. These actors were found several times on the same scenes multiple times.

3.3 Comparison between the graph the preferential attachment network

The preferential attachment network has been created from the Networkx library with the average degree and the size of the network as parameters.
We obtain for the Barabasi-Albert graph an assortativity degree of -0.149. This means that we also have to deal with a dissociative network just like our graph. The application of the Louvain algorithm does not give us much particular information, except the fact that the preferential attachment network presents very diversified communities. This could be explained by its property of heteregenous density of the preferential attachment network. We cannot speak of a community in this case.

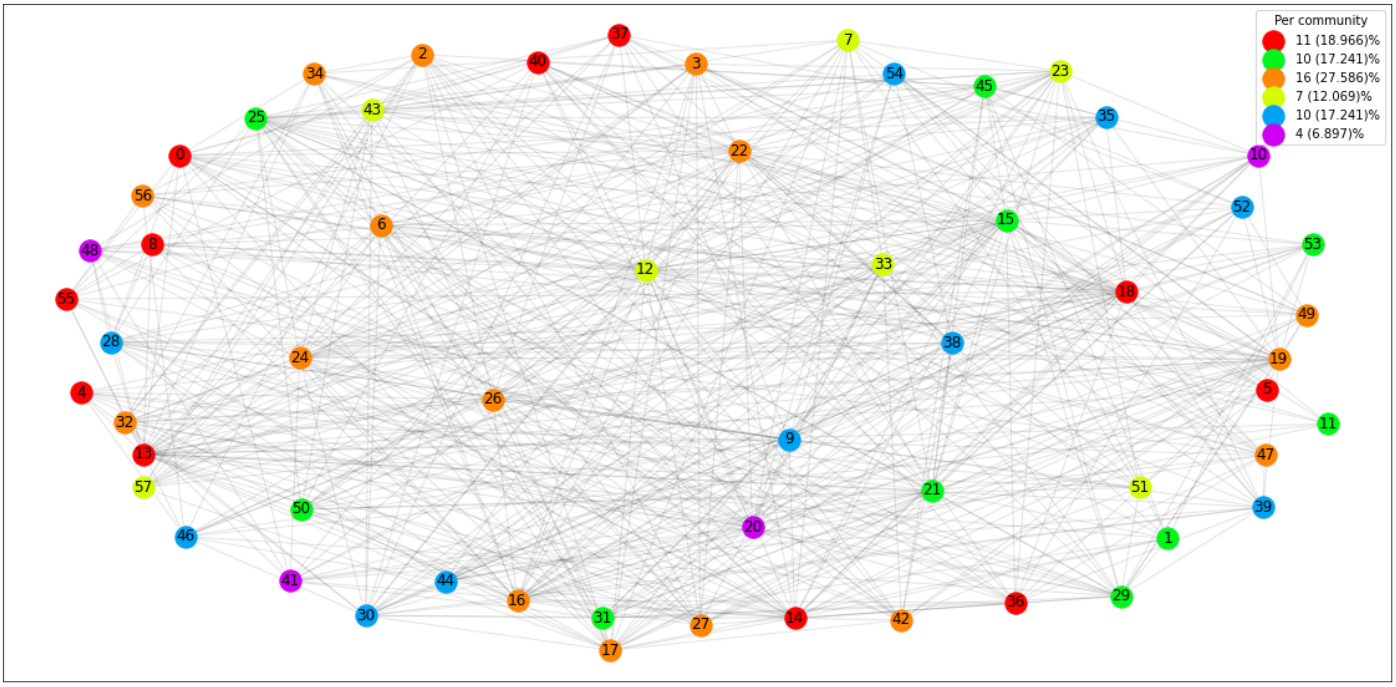

Figure 6: Communities detection result in Barabasi graph

Another important thing to notice is the degree distribution of the two graphs. In our graph, we have many hubs as nodes with less degrees like with the preferential attachment network. These networks have heterogeneous degree distribution.

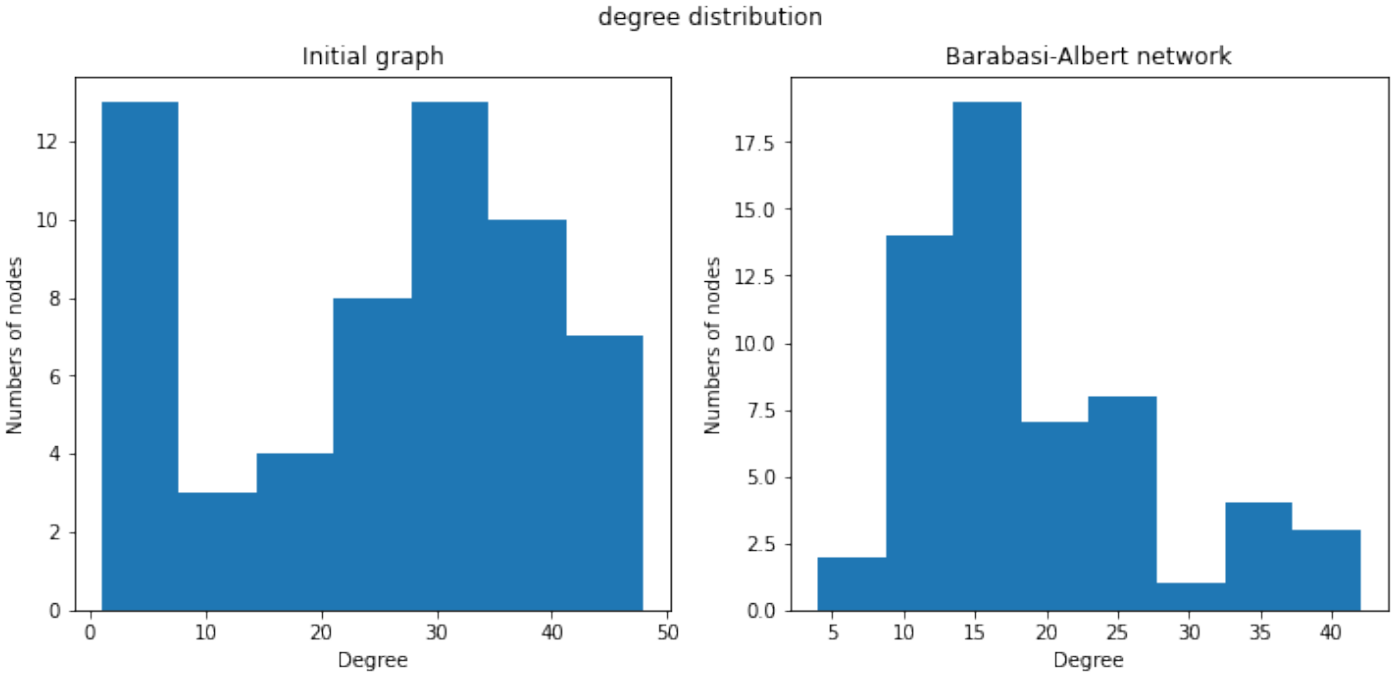

Figure 7: Degree distribution

Based on clustering coefficient distribution, we can also see that our graph has more nodes interconnected between them so more triangles unlike the Barabasi-Albert network.

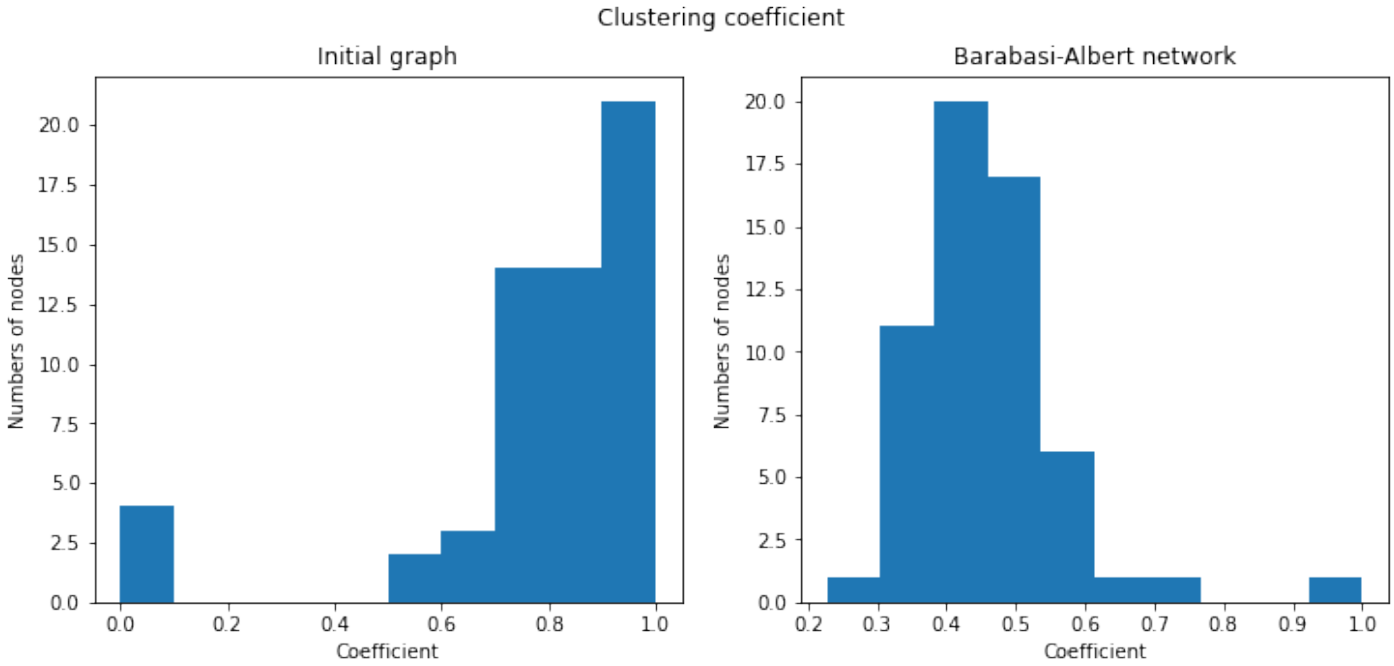

Figure 8: Clustering coefficient

The nodes of our graph have clustering coefficients that are much greater than the Barabasi-Albert model. The construction of this model implies that only a few nodes of the graph, called hubs, are highly connected to the other nodes. The difference between the two graphs is explained by the idea of the movie : all the characters of the Marvel Cinematic Universe are united and therefore, many of them appear in the same scenes. The Barabasi-Albert model thus does not accurately simulate the interactions of the specific movie.

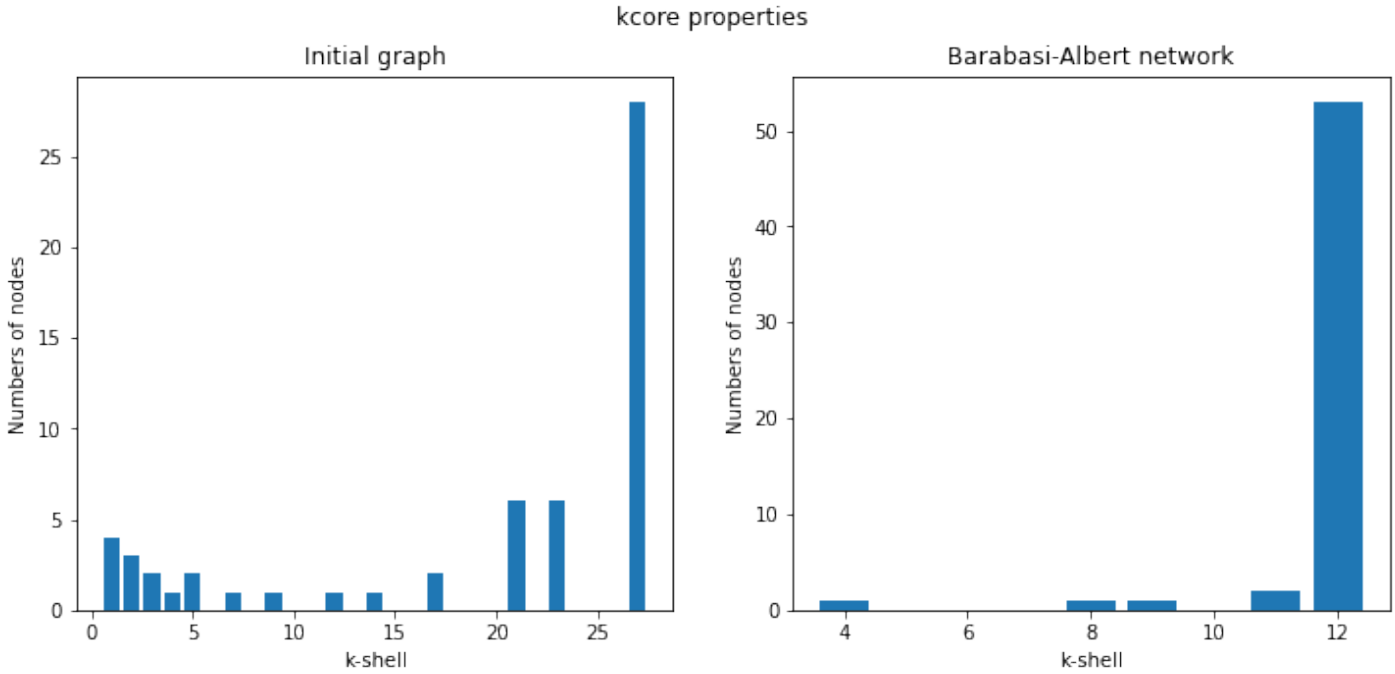

Figure 9: kcore decomposition

4 Influence Maximization problem

We have implemented a function that simulates the spread process using Independent Cascade Model over the graph.This function takes as input a graph, a set of seed nodes, a propagation probability (fixed to 0.1) and the maximum number of simulations we want to do. It returns the average of the activated nodes.

For each node of the set of seed of nodes, we take its neighbors and randomly test their activation according to the propagation probability. For an iteration, we make sure to remove the nodes already used once to launch the epidemic and we add the new contaminated ones. An iteration stops once there are no more nodes likely to contaminate and we repeat the process until the maximum number of iterations. From there, it is possible to have an idea of the average number of contaminated nodes.

For the implementation of the greedy algorithm, for a number K (the size of the set of seed of nodes) fixed at the beginning thanks to a percentage with respect to the nodes of the network, we will first look for the node outside the set of seed of nodes having the most impact for the contamination. We want to specify that the set of seed of nodes is initialized as an empty array.
Progressively, we will first take the node with more influence, to add the neighbor improving the average of the contaminated nodes and the neighbor of the neighbor and so on... We will stop at iteration K.

Each time we take care to save the best results (the most efficient nodes, time taken to carry out each contamination), but also the intermediate values to ensure the representation of the data. The result obtained:

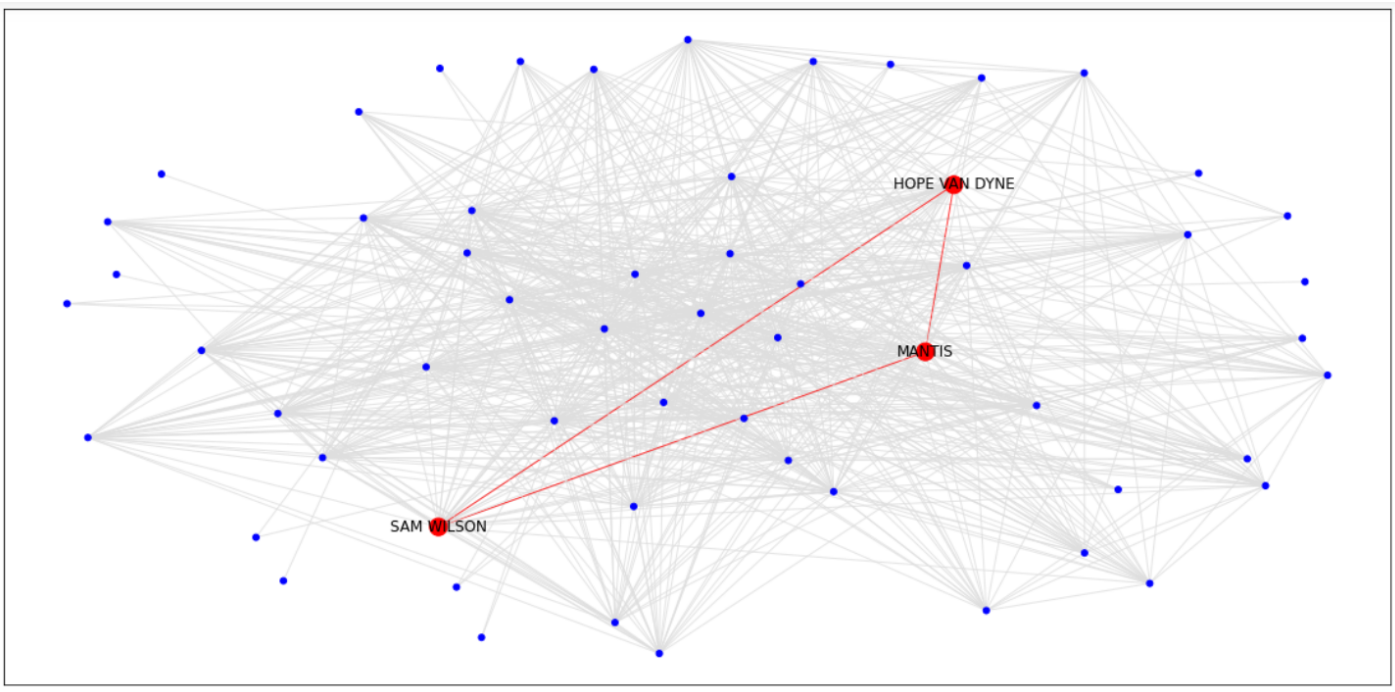

Figure 10: Network representing the set of seed nodes chose by the greedy algorithm

In addition to the spread using the greedy algorithm, we also ran simulations using the nodes with the highest degrees as seed nodes, as well as a random set of nodes.

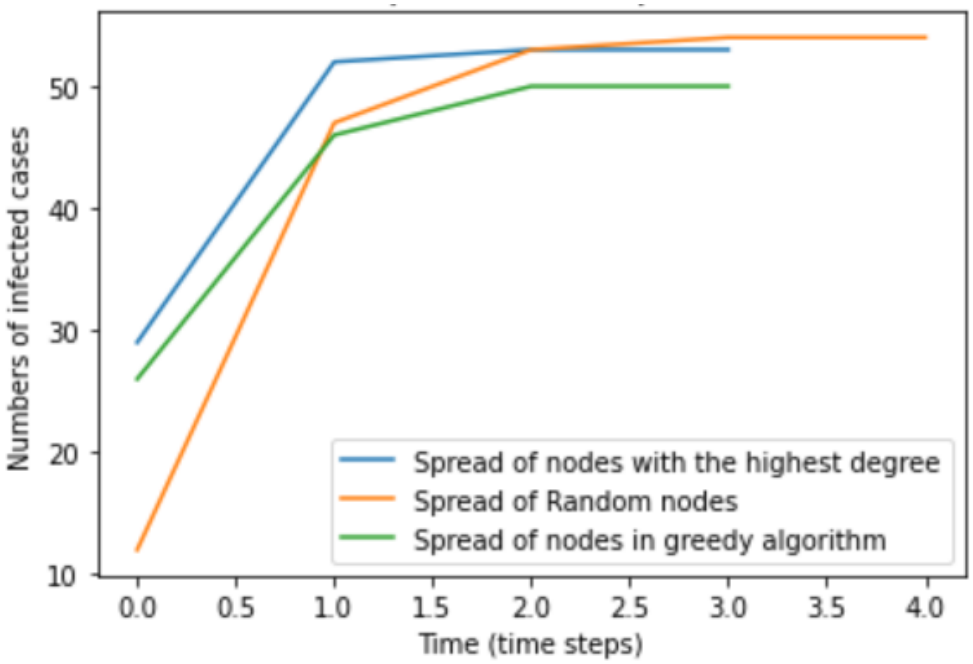

Figure 11: Comparition of spreads

Our graph being well interconnected, using a random set of nodes or the nodes with the highest degrees as seed does not significantly change the total amount of nodes reached by the spread.

The most optimal set of nodes is the one selected by the greedy algorithm. Notice that this set contains nodes with very few edges compared to most of the others. This can be explained by the fact that if a node has few edges, its probability to be reached by the spread is small. This makes them difficult to access. Therefore, starting the spread with these nodes allows it to reach the greatest number, since the highly connected nodes have a far greater probability to be infected.

5 Conclusion

Due to the choice of the movie, most of the nodes of our graph are very interconnected, which gives it properties that are not usually seen in social networks. This has been highlighted by the comparison with the Barabasi-Albert model, that contains far less nodes with a high clustering coefficients, hubs, than our graph. Those hubs are contained in the inner k-shells.

This particular idea of reuniting many characters of the Marvel Cinematic Universe implies that many of them appear in the same scenes, which also makes the isolation of communities complicated.

When applying the greedy algorithm to find the best set of nodes for a spread, we get that the most efficient solution is to start with nodes that are little connected, since they would be less likely to be reached in other cases. Thus, the greedy algorithm finds better solutions than if we use the nodes with the highest degrees or random nodes as the seed set.