The University of Franche-Comté

Master CompuPhys - Network Homework

What if COVID19 epidemic happened in GAME OF THRONES

Chiari Even
Fadlallah Ali
M1 Compuphys 2022

Supervisor : Celestin Coquide

December 11, 2022

Contents

Introduction

In this project, we will study the GOT interaction network (GOT-NET), a graph G(V, E) consisting of a set of nodes V representing the characters from the television series Game of Thrones and a set of links E based on the types of interactions between the characters. We will consider G as a weighted graph, with the weight of each link representing the number of interactions between a pair of characters in a given season. The first part of our analysis will focus on conducting a topological analysis of the network, including an examination of the links and centrality measures of the nodes. In the second part of the project, we will model and analyze the spread of a contagion through the network.

I Topological analysis

In this part we will do a topological analysis of the network, analysis of the links, node's centrality measures and network resilience against attacks.

Ex 1.1

We can group the interactions of GOT-NET into directed and undirected interactions. Directed interactions are those in which the direction of the interaction is specified, while undirected interactions are those in which the direction is not specified.

we can classify the following as directed interactions:

- Character A speaks directly after Character B
- Character A speaks about Character B
- Character C speaks about Character A and Character B

These interactions involve a specified direction, with one character speaking directly after or about another character.

We can classify the following as undirected interactions:

- Character A and Character B are mentioned in the same stage direction
- Character A and Character B appear in a scene together

These interactions do not involve a specified direction, as both characters are mentioned or appear together without specifying who initiated the interaction.

There are several arguments that could be made in favor of considering the GOT-NET as an undirected graph.

One argument is that some interactions in GOT-NET do not involve a specified direction. For example, the interaction of "Character A and Character B are mentioned in the same stage direction" does not involve a specified direction, as both characters are mentioned together without specifying who initiated the interaction. In an undirected graph, the edges between nodes do not have a direction, so this type of interaction would be represented naturally in an undirected graph.

Another argument is that many of the interactions in GOT-NET are symmetrical. For example, the interaction of "Character A speaks directly after Character B" can be considered the same as the interaction of "Character B speaks directly after Character A". In an undirected graph, the edges between nodes are symmetrical, so this type of interaction would be represented naturally in an undirected graph.

A third argument is that the interactions in GOT-NET are likely to be reciprocal, with characters having interactions with each other in both directions. For example, if Character A speaks directly after Character B, it is likely that Character B also speaks directly after Character A at some point. In an undirected graph, the edges between nodes are bidirectional, so this type of reciprocal interaction would be represented naturally in an undirected graph.

Overall, these arguments suggest that GOT-NET could be considered as an undirected graph, as it would allow for the natural representation of the interactions in the network, including the lack of a specified direction, the symmetry of the interactions, and the reciprocal.

Ex 1.2

An adjacency matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of nodes are adjacent or not in the graph. In the special case of a finite simple graph, the adjacency matrix is a (0,1)-matrix with zeros on its diagonal. If the graph is undirected, the adjacency matrix is symmetric.

Ex 2.1

Node centrality is a measure of the importance or influence of a particular node in a graph. Centrality measures are used to identify the most important nodes in a network based on various criteria, such as the number of connections they have, the number of neighbors they have, or the amount of traffic that passes through them. Some common centrality measures include degree centrality, betweenness centrality, and closeness centrality. These measures can be useful for a variety of applications, such as identifying key players in a social network or identifying bottlenecks in a transportation network.

Ex 2.2

The degree of a node, denoted as $D(i)$, is the number of connections or edges that the node has to other nodes in the graph. In other words, it is the number of neighbors that the node has. In general, the degree of a node can be calculated as follows:

$$D(i) = \sum_{j \in E} x_{i,j}$$

Where $x_{i,j} = 1$ if $x_{i,j} \in E$, 0 otherwise

In the context of GOT-NET, the degree of a node would represent the number of interactions that the character had in a particular season. For example, a character might have a high degree in season 1, indicating that they had many interactions with other characters in that season, but a low degree in season 2, indicating that they had fewer interactions in that season.

Ex 2.3

In graph theory, the closeness centrality of a node is a measure of how close or connected the node is to all other nodes in the graph. It is typically calculated as the reciprocal of the sum of the shortest distances between the node and all other nodes in the graph. In other words, a node with a high closeness centrality is one that can reach many other nodes quickly, while a node with a low closeness centrality is one that is more isolated and cannot reach other nodes as easily.

In the context of GOT-NET, the closeness centrality of a node would represent how close or connected a particular character is to all other characters in a particular season. For example, a character with a high closeness centrality in season 1 might be one that has interacted with many other characters and can reach them easily, while a character with a low closeness centrality in season 2 might be one that has had fewer interactions and is more isolated.

The formula for calculating the closeness centrality of a node is:

$$C(i) = \frac{n-1}{\sum_{j=1}^{n} d(i,j)}$$

where n is the total number of nodes in the graph, $d(i,j)$ is the shortest distance between nodes i and j, and the sum is over all nodes j.

Ex 2.4

In graph theory, the betweenness centrality of a node is a measure of how important or influential the node is in terms of the flow of information or traffic through the graph. It is typically calculated as the number of shortest paths between pairs of nodes that pass through the node. In other words, a node with a high betweenness centrality is one that lies on many of the shortest paths between other nodes, while a node with a low betweenness centrality is one that lies on fewer of the shortest paths.

In the context of GOT-NET, the betweenness centrality of a node would represent how important or influential a particular character is in terms of the flow of interactions between other characters in a particular season. For example,

a character with a high betweenness centrality in season 1 might be one that is involved in many of the interactions between other characters and plays a central role in the network, while a character with a low betweenness centrality in season 2 might be one that is less involved in the interactions and plays a more peripheral role.

The formula for calculating the betweenness centrality of a node is:

$$B(i) = \sum_{s,t \in V} \frac{\sigma_{st}(i)}{\sigma_{st}}$$

where V is the set of nodes in the graph, σ_{st} is the total number of shortest paths between nodes s and t, and $\sigma_{st}(i)$ is the number of those shortest paths that pass through node i. The sum is over all pairs of nodes s and t.

Ex 2.5

The 5 most important characters for each season in terms of degree, closeness and betweenness centrality measures, are shown in the tables below.

Season	Main character 1	Main character 2	Main character 3	Main character 4	Main character 5
1	NED	TYRION	ROBERT	CATELYN	ROBB
2	JOFFREY	TYRION	CERSEI	ROBB	ARYA
3	ROBB	TYWIN	TYRION	CATELYN	JOFFREY
4	JOFFREY	TYRION	CERSEI	SANSA	JON
5	CERSEI	JON	LITTLEFINGER	STANNIS	SANSA
6	SANSA	JON	CERSEI	TYRION	JAIME
7	JON	DAENERYS	TYRION	CERSEI	DAVOS
8	SAM	DAENERYS	TYRION	SANSA	ARYA

Table 1: Main Characters in terms of degree

Season	Main character 1	Main character 2	Main character 3	Main character 4	Main character 5
1	NED	ROBERT	CATELYN	TYRION	JON
2	TYRION	JOFFREY	NED	ROBB	CERSEI
3	ROBB	NED	CATELYN	TYWIN	JOFFREY
4	JOFFREY	JAIME	NED	CERSEI	TYRION
5	CERSEI	LITTLEFINGER	STANNIS	SANSA	ROOSE_BOLTON
6	SANSA	TYRION	JON	CERSEI	JAIME
7	JON	TYRION	DAENERYS	CERSEI	DAVOS
8	SAM	DAENERYS	TYRION	DAVOS	SANSA

Table 2: Main Characters in terms of closeness

Season	Main character 1	Main character 2	Main character 3	Main character 4	Main character 5
1	NED	TYRION	CATELYN	ROBERT	DAENERYS
2	ARYA	TYRION	ROBB	JON	NED
3	ROBB	NED	ROBERT	BRAN	JON
4	JOFFREY	NED	STANNIS	JON	VARYS
5	STANNIS	CERSEI	LITTLEFINGER	JON	SANSA
6	SANSA	JON	TYRION	JAIME	CERSEI
7	JON	DAENERYS	CERSEI	SAM	SANSA
8	DAENERYS	SAM	ARYA	BRONN	TYRION

Table 3: Main Characters in terms of betweenness

All of this table are in the exercise 2 folder with the values in .txt files

Ex 2.6

The files that contains the list of GOT's character with their associated centrality scores for each season, can be found int the exercise 2 folder under the name GOT_stats_Si, where i the season number.

In the season five the data contains an error with the character WOLKAN, in the character we have "MAESTER _WOLKAN" an "WOLKAN" this is the same character but appear two time in the list of node, in the edges list "MAESTER _WOLKAN" is mentionned for the interaction but "WOLKAN" is not so we can't compute the degree, the closeness and the betweenness.

This error produce a 'None' value in the txt file. However, It barely affect our calculations. We didn't want too modify the data so we kept that value.

In order to finish the first exercise 1 we have plot all the graph to have a representation of each season.
We have scale the nodes sizes, the font size with the degree of each nodes.
We scale the width of links with the weight of each interactions and we have attributes the colors of each nodes with the same colors in function of their house.

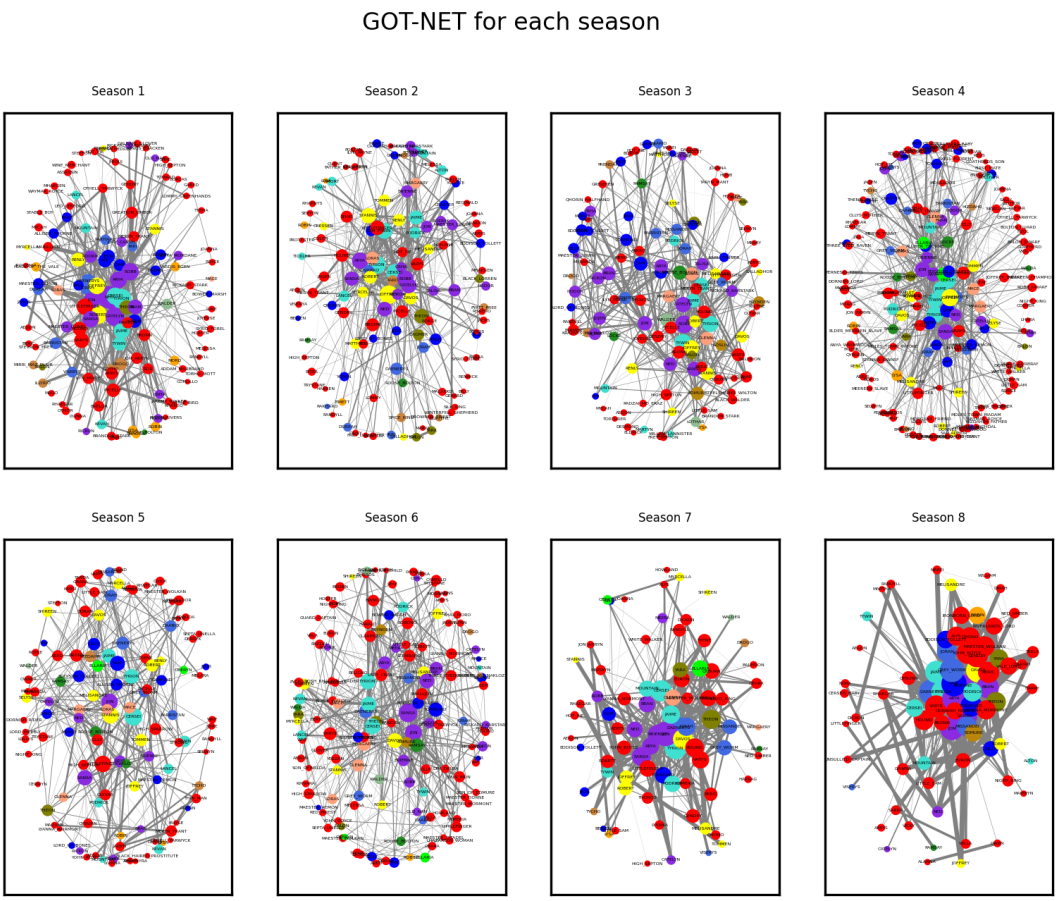

Figure 1: GOT-NET for each season

Ex 3.1

The weighted degree of a node in the GOT-NET represent the sum of the weights of the interactions that the character represented by the node has with other characters in a particular season. For example, if we assign higher weights to interactions that involve a character speaking directly after another character or appearing in a scene together, the weighted degree of a node would represent the sum of the weights of these more significant interactions that the character has with other characters.

The specific weights assigned to the edges in GOT-NET would determine the exact meaning of the weighted degree for each node. However, in general, This measure could be useful for quantifying the importance or influence of a character in terms of the interactions they have with other characters in a particular season of the show.

Ex 3.2

In the program this table are in the folder exercise 3 with the values.

Season	Main character 1	Main character 2	Main character 3	Main character 4	Main character 5
1	NED	TYRION	CATELYN	ROBERT	JON
2	TYRION	CERSEI	ARYA	THEON	JOFFREY
3	TYRION	ROBB	TYWIN	SANSA	CERSEI
4	TYRION	CERSEI	JAIME	TYWIN	JON
5	CERSEI	JON	TYRION	DAENERYS	STANNIS
6	JON	SANSA	TYRION	JAIME	DAVOS
7	JON	DAENERYS	TYRION	CERSEI	SANSA
8	TYRION	JON	DAENERYS	JAIME	SANSA

Table 4: Main Characters in terms of weighted degree

Ex 3.3

The files that contains the list of GOT's characters with their associated weighted degree for each season, can be found int the exercise 3 folder under the name Char_weighted_degree_Si, where i the season number.

Ex 3.4

Each interaction is weighted in the data, this weight measure how many times two character have an interaction.With this graph we can in-light the most important character and their interactions so we can plot the distribution of the weight of each links with the frequency of it and the weight on the x-axis for each season :

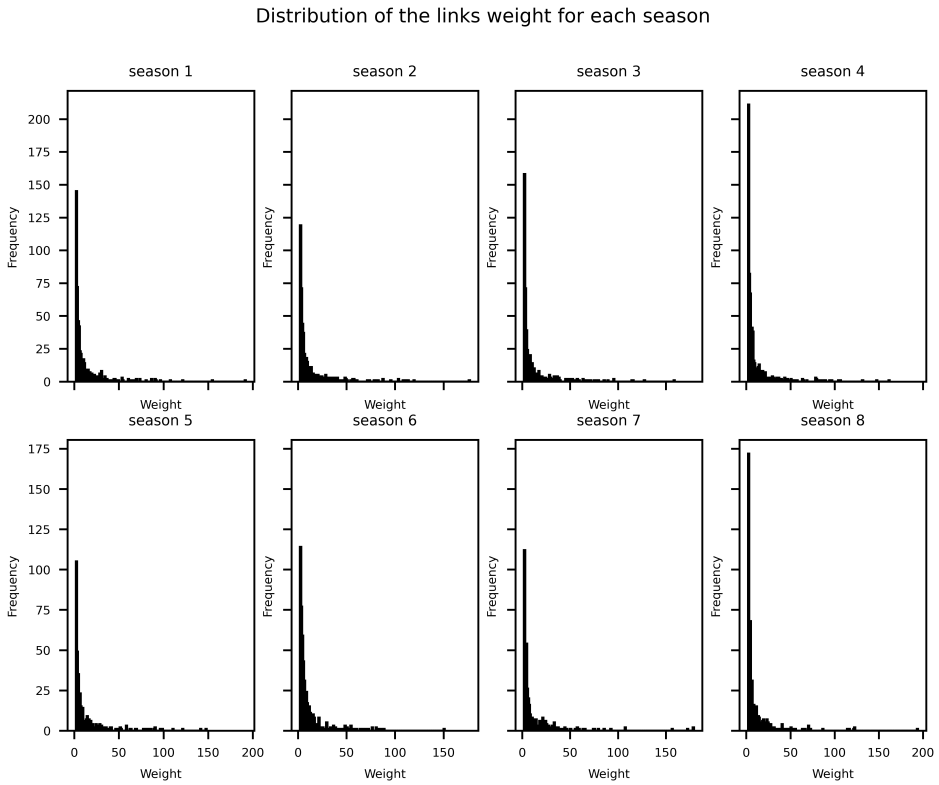

Figure 2: Distribution of the links weight for each season

This graph (clearly more readable in the program with the zoom) show the number of little interactions (between 2 and 15 for the weight and the highest frequency) are the most many type of interactions and few's big interaction for the weakest frequency, this make sens because the weightiest interaction are for the most important character (5 or 10 character for each season).

Ex 4.1

The connected_components() function is a part of the networkx package in Python. This function returns the connected components in a graph.

The input to this function is a graph, which can be represented using a dictionary in Python. The keys of the dictionary are the nodes of the graph, and the values are the lists of nodes that are directly connected to the key node.

The output of this function is a list of sets. Each set in the list contains the nodes that belong to a connected component in the graph.

Ex 4.2

Season	Connected Components	Relative Size
1	1	1
2	2	0.89, 0.11
3	1	1
4	2	0.9825, 0.0175
5	1	1
6	2	0.9437, 0.0563
7	1	1
8	2	0.973, 0.027

Table 5: Connected Components and their Relative Size

This table is the exercise 4 folder in a .txt file.

Ex 4.3

The seasons 2, 4, 6 and 8 have 2 connected components. Below we draw the network corresponding to the smallest connected components of those seasons.

For season 2 we can see that the interactions are happening between 3 factions, the Targaryen (royalblue), Westoros (red) and Essos (peru), and that Daenerys is the main character in this stroyline. This is consistent with the show, since in season 2 there was a plot about Daenerys, and it was happening in Essos which is a different continents from the one the main plot was happening, Westoros.

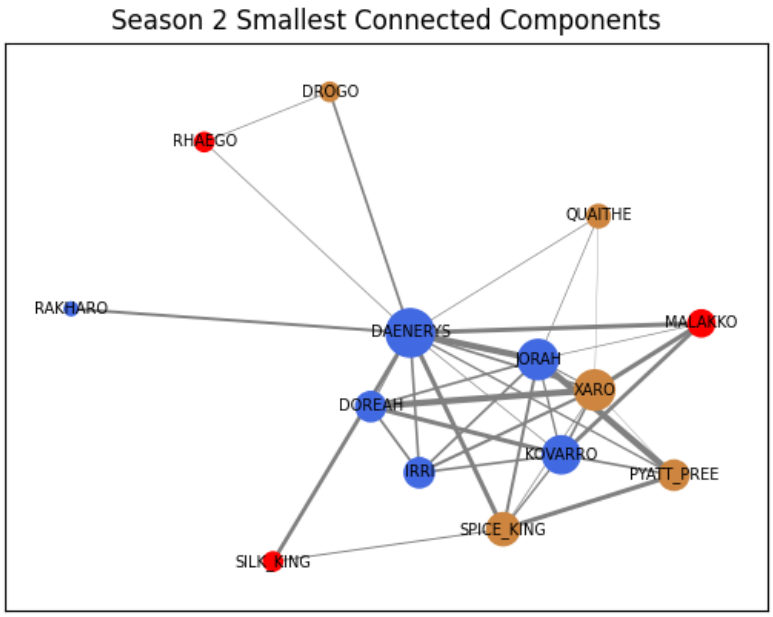

Figure 3: Smallest subgraph for the season 2

In season 4, 3 characters belonging to Westoros interact only with each other.

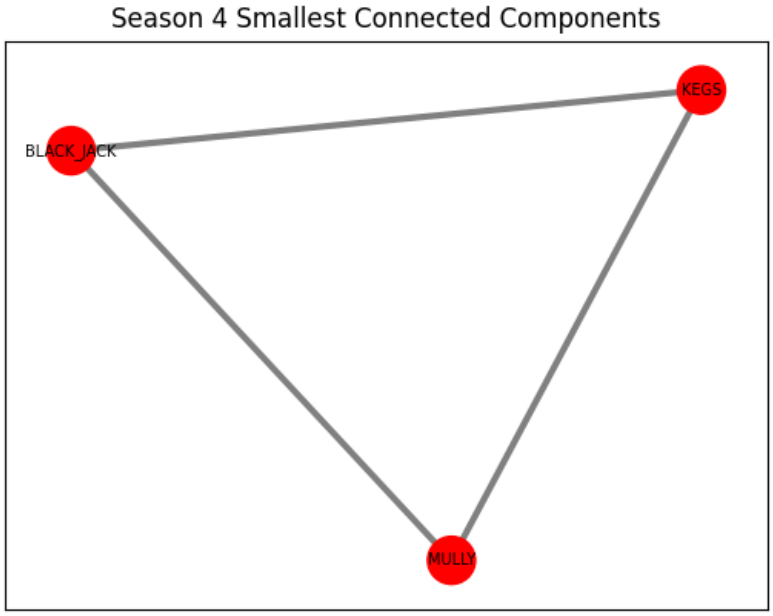

Figure 4: Smallest subgraph for the season 4

Similarly in season 6, 8 characters belonging to Westoros interact only with each other.

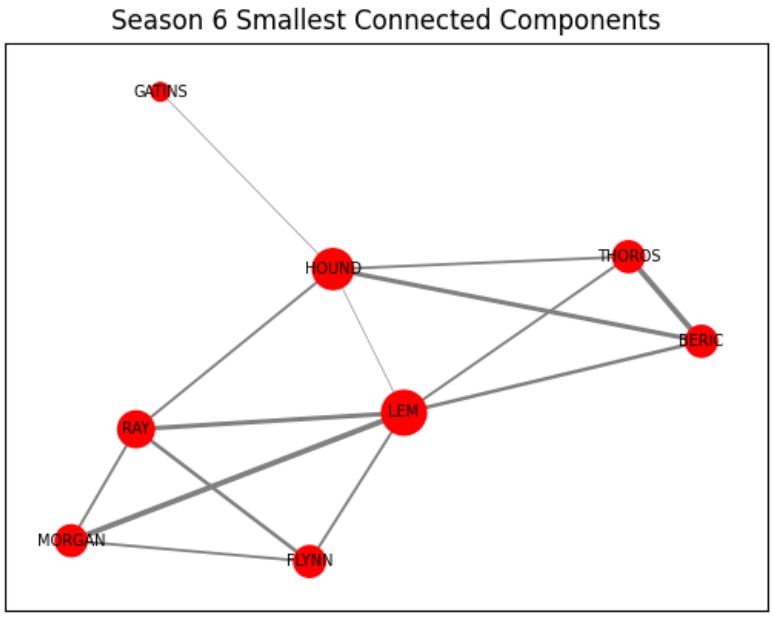

Figure 5: Smallest subgraph for the season 6

In season 8, Littelfinger (now associated Westoros) and Ramsay (Frey) interact only with each other.

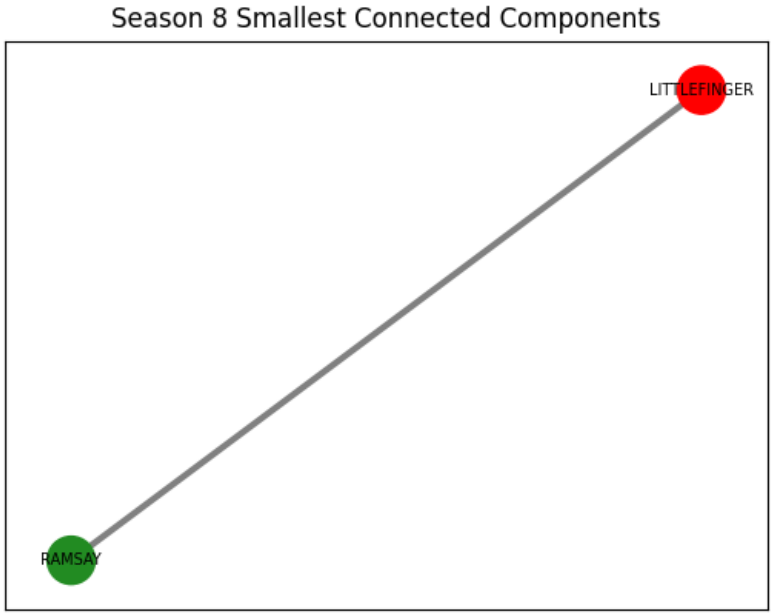

Figure 6: Smallest subgraph for the season 8

Ex 4.4

Below we plot the evolution of the average size of connected components |c| with the fraction of attacked nodes f, where f evolves in the range [0, 1] with a step df = 0.05

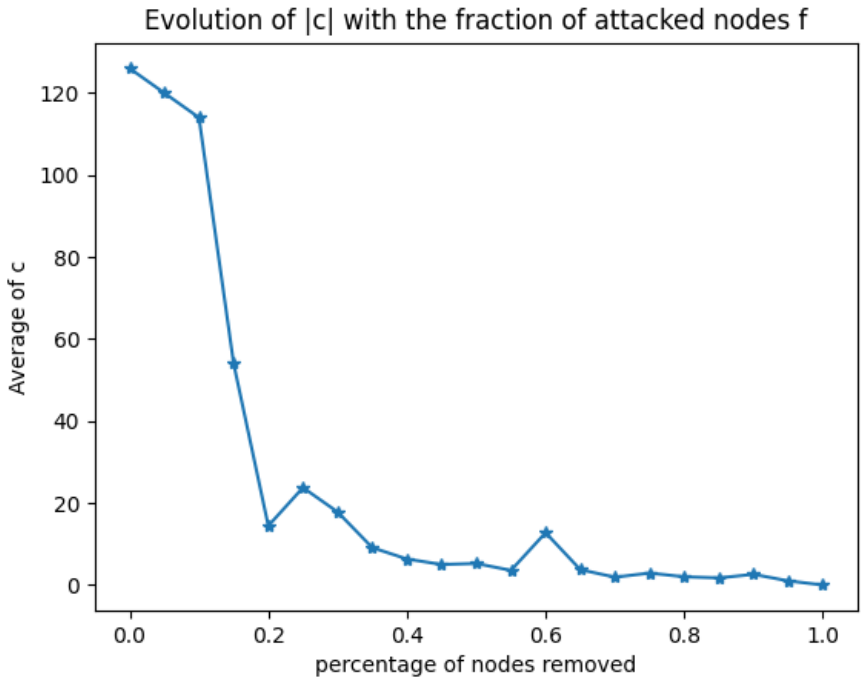

Figure 7: Evolution of the average of c with f of the season one

Ex 4.5

We can observe a transition where the average size of the connected components drops significantly after a certain fractions of the nodes are removed. In this plot the critical fraction is 15%, however, we note that this evolution is not stable, when we run to code multiple times the f_c fluctuate, between 5, 10 and 15 % .

Ex 4.6

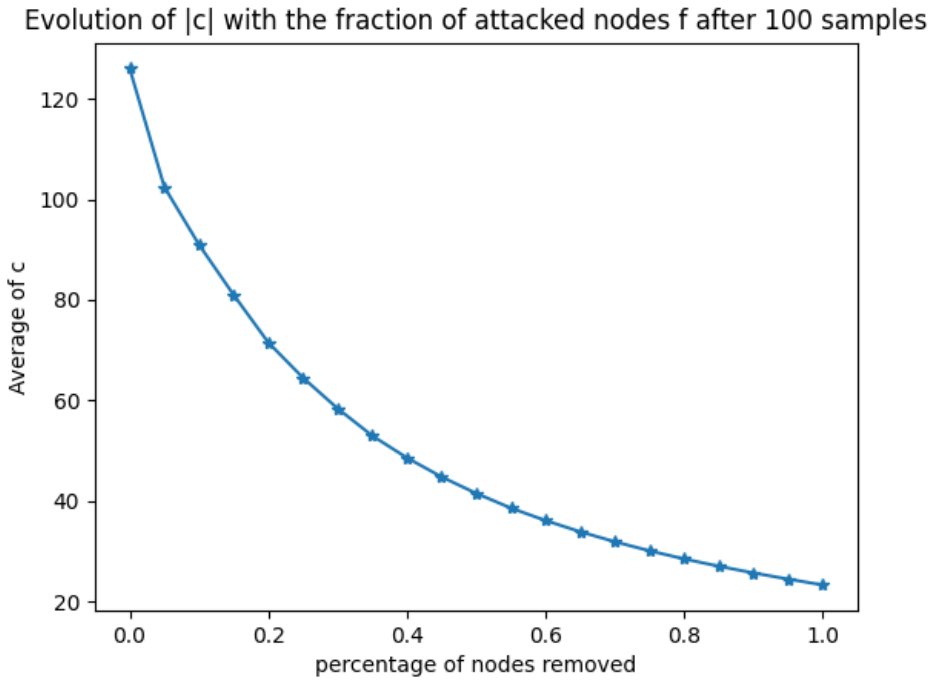

Figure 8: Evolution of the average of c with f for 100 samples of the season one

Ex 4.7

The average connected component size decreases with node removals. However, it appears to be smoothly decreasing. There is no clear transition from a connected phase to a disconnected phase as the fraction of removed nodes increases.

Ex 4.8

The average connected component size is a measure of the connectivity of the network. As the average connected component size decreases as a result of node removal, it indicates that the network is becoming more fragmented and less connected and the connected components are smaller on average.

This indicates that the network may not be very resilient to random node attacks, as the removal of even a small number of nodes can significantly decrease the average connected component size and fragment the network.

II Susceptible-Infectious-Susceptible (SIS) Model

In this section, we are interested in modeling the spread of a pandemic, such as COVID-19, in the universe of the Game of Thrones (GOT). To do this, we are using the SIS model, which is one of the simplest contagion models.

Ex 1.1

Using the SIS Model, we simulate the epidemic in the GOT world. To do this we implemented Algorithm I in a function called covided(), which simulate an initial number of infected nodes, store these initial values in a file. Then it apply the algorithm I to simulate the epidemic. At the end we get a dictionary containing the nodes and their final states (infected 1, susceptible 0), a dictionary of the nodes and a number representing how many time each got infected, and finally a list of the number of infected nodes at each time step. We then create a file to store how many times each nodes got infected

Ex 1.2

for the 3 set of parameters we simulate the epidemic 1000 time, each time we a random set of initial nodes are infected. We get the following

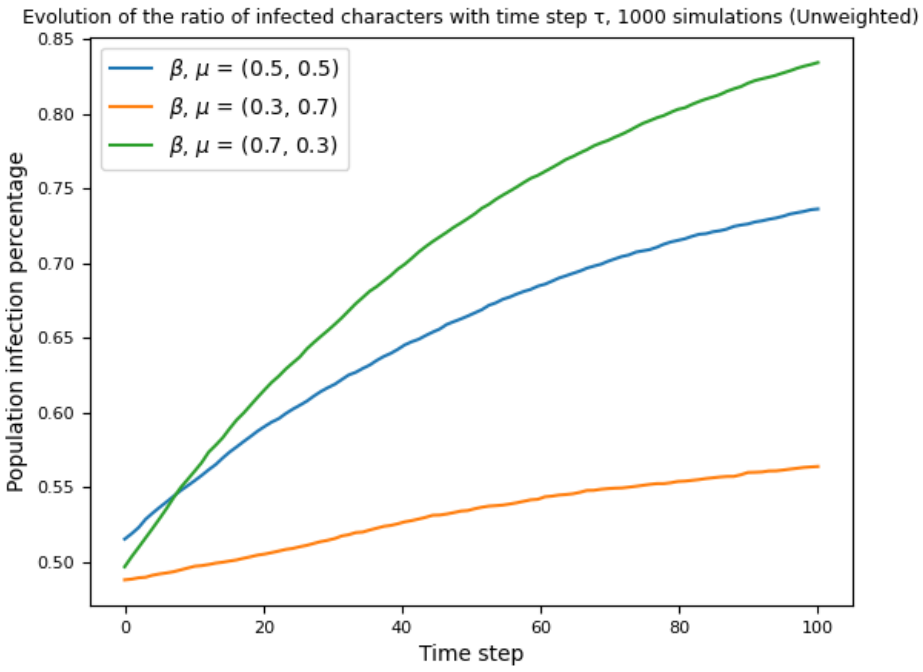

Figure 9: Evolution of the ratio of infected characters with time step (Unweighted)

We can see that the initial percentage of infected nodes has an average value of 50%, which is expected since we used random.random() to generate the initial percentage, thus after 1000 simulation it's logical to get 50%.

for an initial percentage of infected characters 50%, we can see that the (0.3, 0.8) parameter is the most steady spread of the virus, since there is a 70% for the infected characters to recover. for (0.5, 0.5) the more rapidly even tho there is 50% for recovery, and for the (0.7, 0.3) the epidemic is quickly spreading.

Ex 1.3

To show a different perspective, we will simulate the epidemic for 2 initial percentages, 80% and 10 %

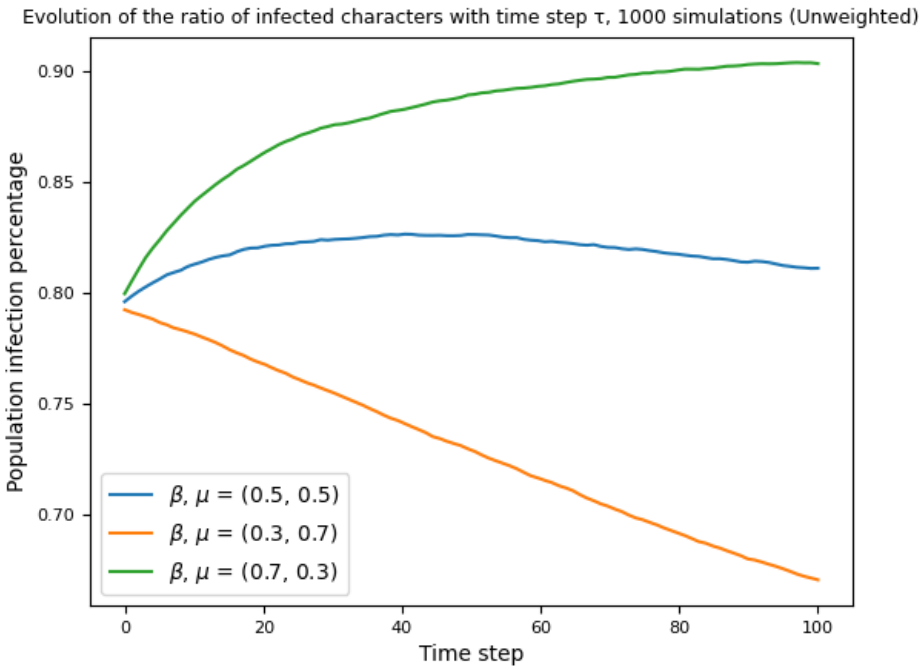

Figure 10: Evolution of the ratio of infected characters with time step f 80 (Unweighted)

for a high initial infection percentage (80 %), We can see that for (0.5,0.5) the virus is spreading at a steady rate. for (0.7, 0.3) he virus is spreading more rapidly than in the first case. And for (0.3, 0.7) the ratio of infected characters is decreasing over time, he disease is being contained more effectively than it is spreading.

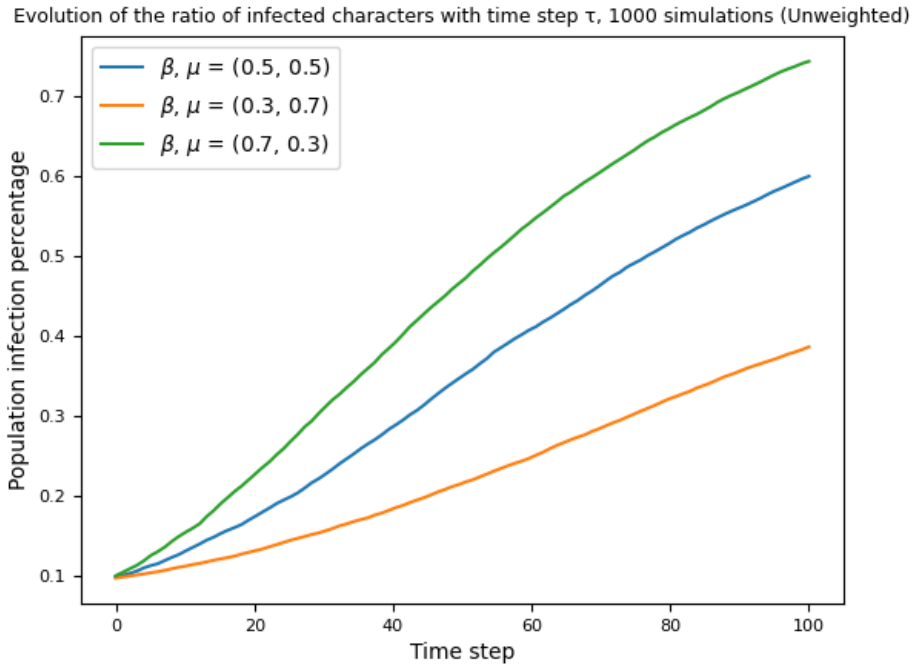

Figure 11: Evolution of the ratio of infected characters with time step f 10 (Unweighted)

for a low initial infection percentage (10 %), We can see that for the 3 cases, the epidemic indeed spread.

Ex 2.1 & 2.2

We add an argument weighted to our covided() function. The function now do the same as previously while working with the weighted network.

Ex 2.3

Similar to what we did in the unweighted network, we simulate the epidemic 1000 time, each time a random set of initial nodes are infected, for the 3 set of parameters. We get the following

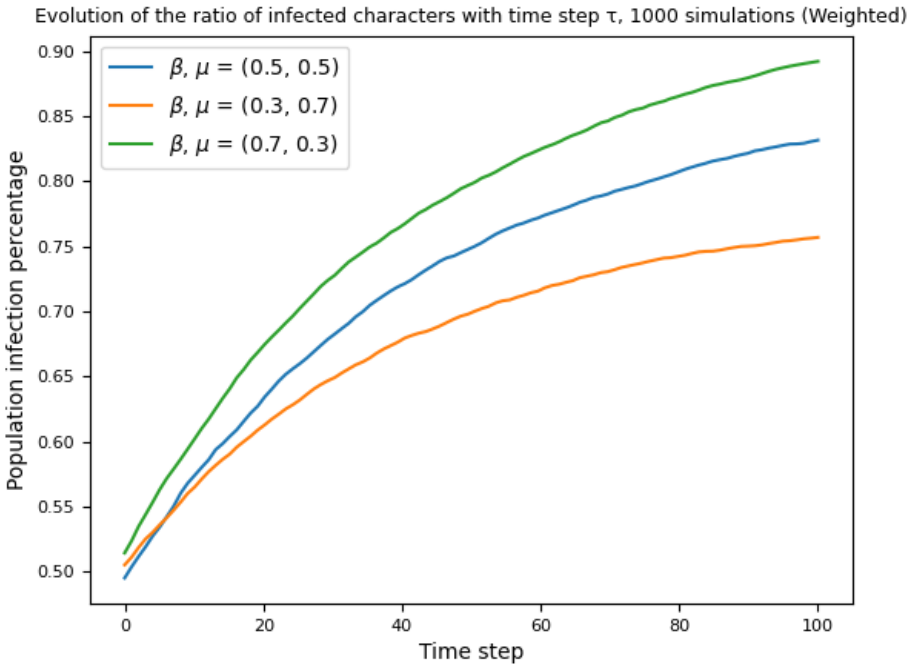

Figure 12: Evolution of the ratio of infected characters with time step (Weighted)

The initial percentage of infected characters is also 50%, we can see that for (0.7, 0.3) and (0.5, 0.5) the epidemic is spreading at similar rates to that of the unweighted network. However, for (0.3, 0.7) we can see that the epidemic is spreading at fast rate, in fact faster than its equivalent in the unweighted degree, where it had a steady spread. This can be attributed to our covided() function, where when we check if node will infect one of its neighbors, we are looping over the number of interactions between these 2 nodes, where previously we only checked once.

Below we plot the evolution for an initial percentage of 80 and 10.

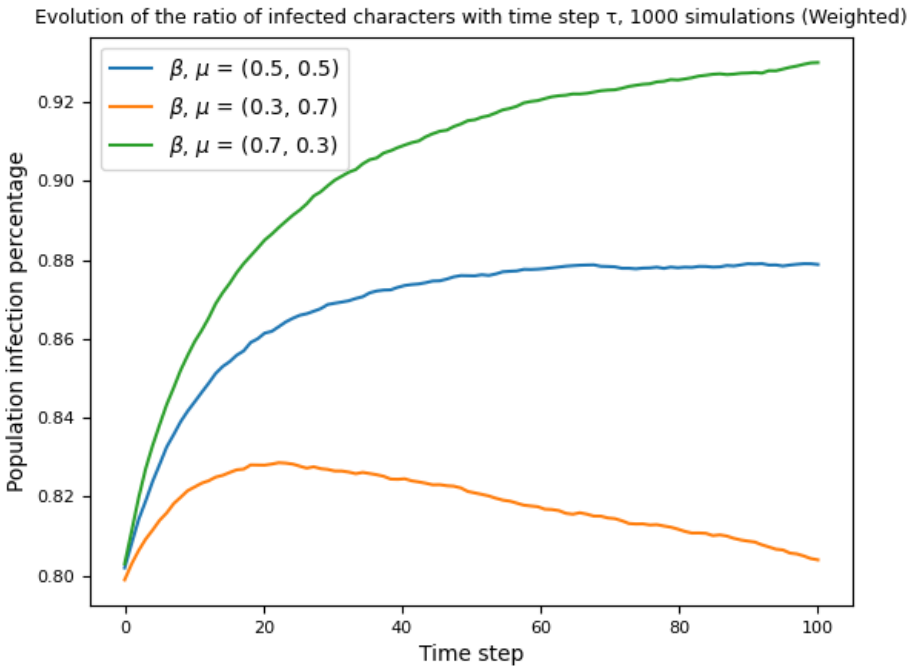

Figure 13: Evolution of the ratio of infected characters with time step f 80 (Weighted)

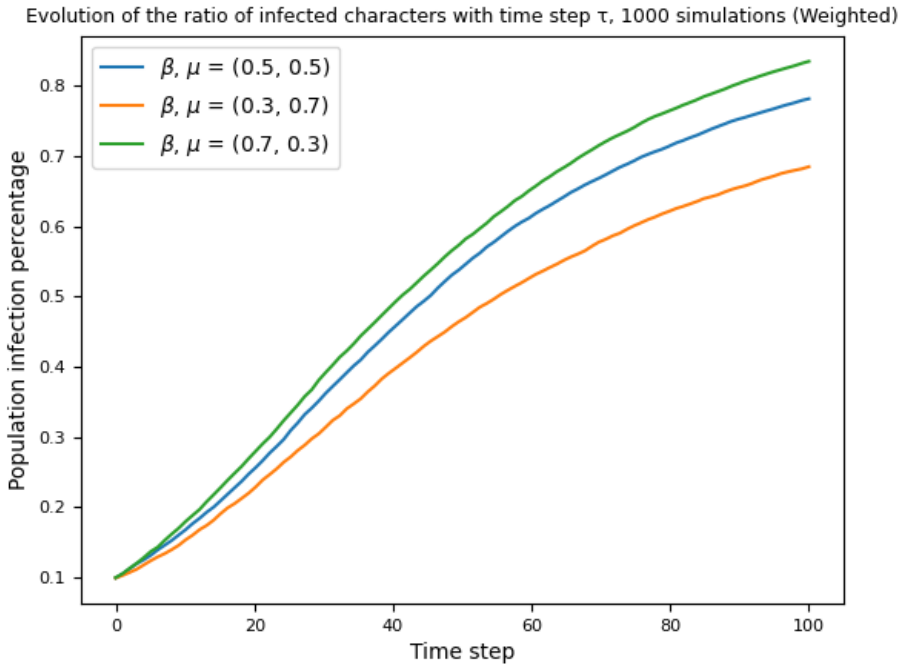

Figure 14: Evolution of the ratio of infected characters f 10 with time step (Weighted)

We can see the similarly to the unweighted network, with the parameter (0.3, 0.7) having a higher spreading rate than it's equivalent in the unweighted network.

III Conclusion

This study was very interesting with all the statistic computation that we have made, the degree, the closeness, the betweeness and the weighted degree are good tools to understand a season and the simulation of covid was very interesting with the SIS model.

List of Figures