Practical Exercise 4: Epidemic spreading on 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

May 05, 2024

Contents

4

Epidemic spreading on complex Networks

1 | Introduction

Modeling the spread of epidemics in complex networks is an important area of research that combines the disciplines of epidemiology, network theory, sociology, mathematics, etc. This field has gained relevance in recent years, particularly with the global spread of COVID-19, emphasizing the importance and urgency of understanding how diseases spread through complex networks.

The complexity of these networks is challenging for modeling because of their heterogeneous structure with different connectivity patterns, which cannot be solved by analytical solutions, due to the diversity of nodes and edges that constitute these networks. Networks can exhibit several characteristics, such as scale-free degree distributions, where some nodes have significantly more connections than others. Also, the temporal nature where connections can change over time as individuals respond to the epidemic, adds another layer of complexity to accurate modeling.

Different models are commonly used to study the spread of disease in a network. The Susceptible-Infected-Susceptible (SIS) and Susceptible-Infected-Recovered (SIR) models are among the primary frameworks. In the SIS model, individuals can transition from susceptible to infected and then back to susceptible, potentially oscillating indefinitely. This model is particularly relevant for diseases where recovery does not provide permanent immunity. While these models oversimplify real-world dynamics to some extent, they provide important insights.

These models are not only central to disease management, but also have broader applications in understanding phenomena that follow similar patterns of spread, such as information dissemination, marketing campaigns, the spread of fake news, and even financial crises, where the contagion model of spread can often be applied. For example, viral marketing strategies often rely on the principles of epidemic spread to maximize the reach and influence of advertising campaigns. In financial markets, epidemic spread models have been used to understand and predict how crises can propagate through interconnected financial institutions, leading to systemic risks.

In summary, understanding how to model the spread of epidemics in complex networks - understanding the various parameters and their effects, and learning how to analyze them - provides us with powerful tools to address a wide range of real-world challenges beyond the public health domain. The insights gained from such studies are critical not only for developing effective disease control strategies, but for addressing issues of communication, marketing, and financial stability.

2 | Methodology

2.1 | General Description

During the previous assignments, we have been swapping the implementation between Python and Julia languages. Therefore we have had the opportunity to try both languages and compare them identifying pros and cons for each of them. For the current assignment, we decided to implement it in both languages. On one hand, Python language is straightforward with a lot of already built packages and a wider number of functionalities on networks and we are more used to its use. On the other hand, Julia could be faster in the execution time, which can be a good fit for the high computation cost for the Montecarlo simulations.

For both languages since there are multiple parameters that affect the outcome, we had to implement two things: fix some of the parameters in order to isolate the effect of the ones that we want to analyze, and also limit the variation of the parameter values that we want to analyze.

The complete parameters are as follows:

For the network:

- Type of network. Variable. Values: Barabasi-Albert, CM Power-Law, Erdos-Renyi, Real networks
- Number of nodes (N). Variable. Values: 500, 600, 700.

Since we do not want to have a large number of different networks, we selected different combinations of the parameters resulting in a total of 17 different networks, where are 14 generated networks and 3 real ones. They are presented in more detail in the Table 4.1 with all their descriptors.

For the SIS model:

- Spontaneous recovery probability (µ). Variable. Values: 0.1, 0.5, 0.9.
- Infection probability of a susceptible (S) individual when it is contacted by an infected (I) one. Variable. Values: 0-1 with a step of 0.02.

For the Montecarlo simulation:

- Initial fraction of infected nodes (ρ_0.). Fixed. Value: 0.20.
- Number of repetitions (Nrep). Fixed. Value: 100.

Table 4.1: Networks Numerical Descriptors

Index	# Nodes	# Edged	Dg(min)	Dg(max)	Dg(avg)	ACC	Assort.	APL	Diameter
Barabasi-Albert_50_600_6.net	600	3759	5	83	12.53	0.1006	-0.0476	2.6441	4
Barabasi-Albert_50_600_9_1.net	600	4885	2	83	16.2833	0.0476	-0.0773	2.5534	4
CModel_power-law_700_10_3.0.net	700	652	1	10	1.8629	0.0032	-0.0382	2.5475	9.22E+18
CModel_power-law_700_8_3.0.net	700	623	1	8	1.78	0	0.0374	1.5806	9.22E+18
Erdos-Renyi_500_0.3.net	500	37579	115	182	150.316	0.3008	0.0014	1.6988	2
Erdos-Renyi_500_0.7.net	500	87563	313	380	350.252	0.7019	-0.0058	1.2981	2
Erdos-Renyi_500_1500.net	500	1500	0	14	6	0.0111	-0.0157	3.5753	9.22E+18
Erdos-Renyi_500_25.net	500	25	0	2	0.1	0	-0.0417	-0.9996	9.22E+18
Erdos-Renyi_500_350.net	500	350	0	5	1.4	0	0.001	3.1984	9.22E+18
Erdos-Renyi_500_50.net	500	50	0	2	0.2	0	-0.087	-0.9991	9.22E+18
Erdos-Renyi_700_0.3.net	700	73519	166	245	210.0543	0.3006	-0.0064	1.6995	2
Erdos-Renyi_700_0.7.net	700	171238	452	526	489.2514	0.6999	-0.004	1.3001	2
Erdos-Renyi_700_40.net	700	40	0	2	0.1143	0	-0.1429	-0.9996	9.22E+18
Erdos-Renyi_700_80.net	700	80	0	3	0.2286	0	-0.1057	-0.9987	9.22E+18
airports_UW.net	3618	14142	1	250	7.8176	0.4957	0.0462	4.4396	17
dolphins.net	62	159	1	12	5.129	0.259	-0.0436	3.357	8
zachary_unwh.net	34	78	1	17	4.5882	0.5706	-0.4756	2.4082	5

- Maximum number of time steps of each simulation (Tmax). Fixed. Value: 1000.
- Number of steps of the transitory (Ttrans) Fixed. Value: 900

As can be seen, all the values that are for the Montecarlo simulations are fixed. This will allow us to have comparable results from the simulation step by setting the same conditions and leaving the important part to analyzing the different values for the SIS model and the network.

With this in mind, we will plot a main graph that will give us the ability to compare these 3 main parameters (network, μ, β) and will allow us to analyze the variation of these in the next section.

Main plot: Behavior of average ρ for each value of β, for multiple values μ.

This means that we will plot the average of averages ρ (average ratio of infected nodes for the stationary phase for all simulations), according to the given value β. And we will introduce with different colors the 3 different values for μ. As it was observed in the example plot extracted from our assignment.

This plot includes both parameter β and μ on the graph, with the main result we are interested in: ρ therefore all of the epidemic spreading simulation data is summarized in this plot, having just one plot for each network.

Auxiliary plot: Behavior of ρ for each time step with multiple sets of β.

This plot shows the relationship between these 3 elements (ρ , time step, β), the different possible values for μ can not be displayed as well, and just one has to be selected. In this case, we will be only showing 1 fixed value which will be μ=0.5 for each of the different networks to keep the information manageable and comparable.

Also, since we have 51 possible values of β, analyzing all of them in the same plot would be confusing. For this reason, only certain values will be plotted: (0.1,0.3,0.5,0.7,0.9) The plot incorporated in the instructions of the current assignment was used as a reference.

Since most of the information cannot be summarized in this plot as it can be done in the main plot, we decided that this one would not be as interesting to add as an analysis. But it was created in the initial or discovery part of the development, and some plots are added as examples in the results section.

Finally, just as a reference and to be able not to only model the results but also view the complexity increase on the time for computation, we have added the total time for simulating each of the different networks (including the Monte Carlo simulation). Since this was not the main objective of the assignment, we decided to keep it out, however, it was very important to make some decisions during the development and test of the whole assignment, and it can be found in the code.

2.2 | Python implementation

For the implementation of the code, we decided to create three different files to keep the development as clean and readable as possible.

- **utils.py**
  This file contains different functions that are needed as auxiliary functions to the development and readability of the process, that are not directly related to the SIS model or Montecarlo simulation.
  - **graph_dict**: Reads a network and converts it into a dictionary for easier manipulation on the SIS model and montecarlo simulation.
  - **save_csv**:Saves the different data into a csv file for further analysis if needed.
  - **save_network_to_pajek**: If there is a generated network for a model, it saves it as a .net file.
  - **plot_beta_vs_p**: Plots and saves as a png image the main plot described earlier in this section.
  - **plot_timesteps**: Plots and saves as a png image the auxiliary plot described earlier in this section.
- **monte_carlo.py**
  This file was created as a package where are embedded both the SIS model and the Montecarlo simulation, consisting of 3 methods.

  - **state_zero**: It helps to initiate the different states of the node at the beginning according to the probability p0.
  - **state_t_plus_one**: Replicates the next state t+1 using the SIS model and the values mu and beta to randomly create the next state for each of the nodes according to its degree and the other parameters.
  - **MC**: It runs the state t+1 according to the parameters of the SIS model (1000 steps and 900 steps are the transitory steps). Iterates these processes according to the number of simulations set, saving all the values to be able to plot them as a further step.

- **main.py**
  Here is where the overall workflow happens. The different parameter values are set: the fixed as direct values, and the variables as lists. It has multiple loops to iterate all over the networks, μ, and β values. It is limited to calling the different functions and methods created in the other two files and taking the time execution time for each process and print information out to the terminal to know the actual status.

2.3 | Julia implementation

The codebase consists of three main components interacting to simulate epidemic spreading dynamics on complex networks. The *generate_models.jl* script generates various network models such as Erdos-Renyi, Barabasi-Albert, and Configuration-Model, saving them in Pajek format. It relies on *NetworkProcessing.jl* and *NetworkModels.jl* modules for network processing and model generation. The *DynamicSimulations.jl* module implements the Monte Carlo simulations using the SIS model, defining the model structure and simulation functions. Finally, *run_experiments.jl* contains the execution of simulations on generated networks, records the results, plots the fraction of infected nodes against infection probability, and exports the results to a CSV file.

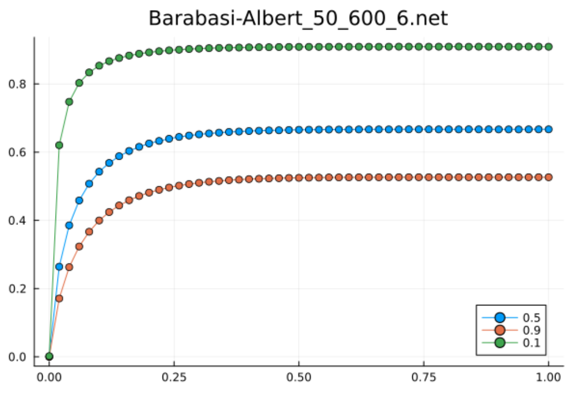
- **generate_models.jl:**
  - *Purpose*: Generates different types of networks according to specified models.
  - *Workflow*:
    - Defines a dictionary *models* containing network models and their parameters.
    - Loops through each model and its parameters, generating networks accordingly.
    - Saves generated networks in Pajek format (.net) and corresponding figures if applicable.
    - Computes and saves experimental and theoretical degree distributions for large networks.

- **DynamicSimulations.jl:**
  - *Purpose*: Implements Monte Carlo simulations of epidemic spreading using the SIS model.
  - *Structure*:
    - Defines an abstract type *Model* and a concrete type *SIS* representing the SIS model.
    - Provides functions for creating SIS models and updating their states.
    - Includes a function *monte_carlo* for running Monte Carlo simulations and calculating the fraction of infected nodes.
    - Defines *allcombinations* function to generate all combinations of parameters.
- **run_experiments.jl:**
  - *Purpose*: Executes Monte Carlo simulations for epidemic spreading on various networks and records the results.
  - *Workflow*:
    - Reads networks from a specified directory.
    - Executes Monte Carlo simulations for each combination of infection and recovery probabilities.
    - Records simulation results including network name, infection probability, recovery probability, and fraction of infected nodes.
    - Plots the fraction of infected nodes against infection probability for each network and saves the plots.
    - Constructs a DataFrame from the simulation results and exports it to a CSV file.

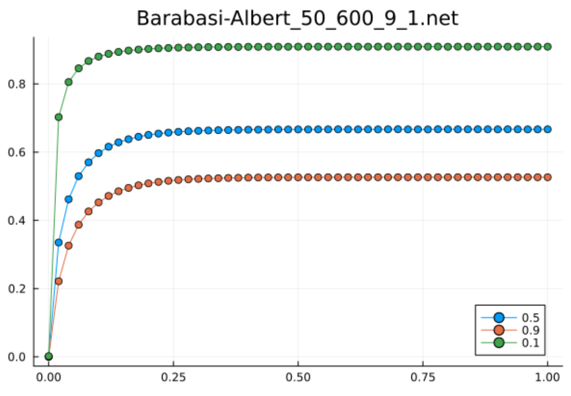
3 | Results & Discussion

One relevant point that we noticed early during the development of the code in both languages was that Julia was outperforming Python in the total execution time. The development of the code was done progressively, starting with small networks with few nodes, and small average degree, few iterations, and time steps, therefore it started being insignificant, however, with the increasing complexity, this total execution time started to seem more relevant.

Therefore, even when we completed the implementation in both programming languages, which are attached to the current report, we decided to complete all the simulations with the proposed networks only with the Julia development.



(a) $BA(50, 600, 6)$

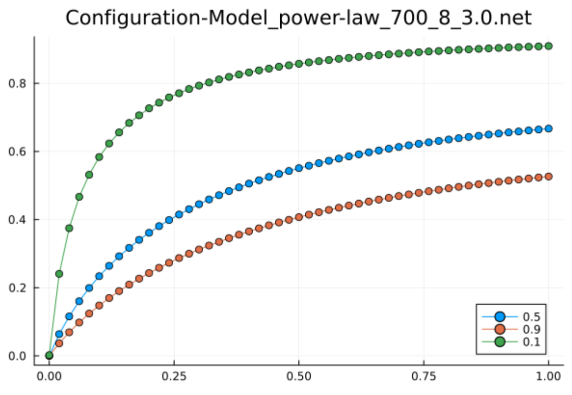


(b) $BA(50, 600, 9)$

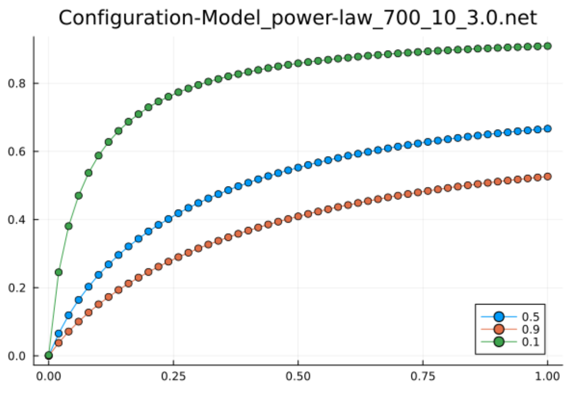
Figure 4.1: Barbasi-Albert Networks plot ρ vs β at different μ values.

Starting with the analysis, we can see in fig 4.1 a comparison between two different generated networks with the Barabasi-Albert model. If we look at the descriptors table, we would notice that they are very similar, with 600 nodes, and even the max degree of both is the same, however, the average increases from 12, for the plot on the left, to 16 for the plot on the right.

This small increase in the average degree has an impact on the behavior where we notice that the value ρ tends to increase faster as the β increases, as well the ρ is higher with the same values of μ.



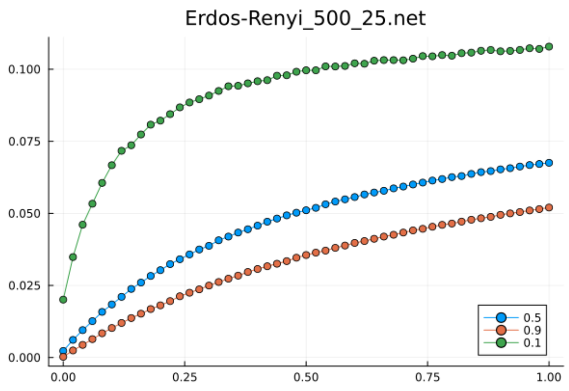
(a) $CM(Plaw, n = 700, gamma = 3.0, max = 8)$



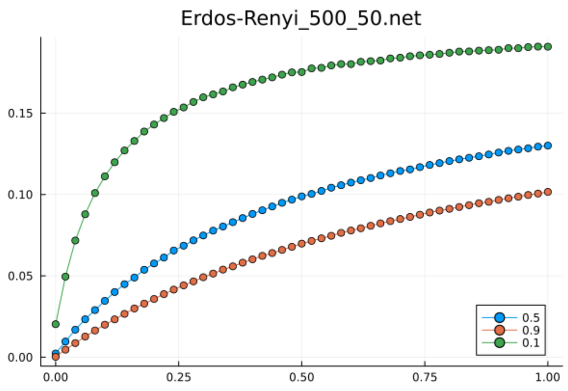
(b) $CM(Plaw, n = 700, gamma = 3.0, max = 10)$

Figure 4.2: Configuration Model (power-law) Networks plot ρ vs β at different μ values.

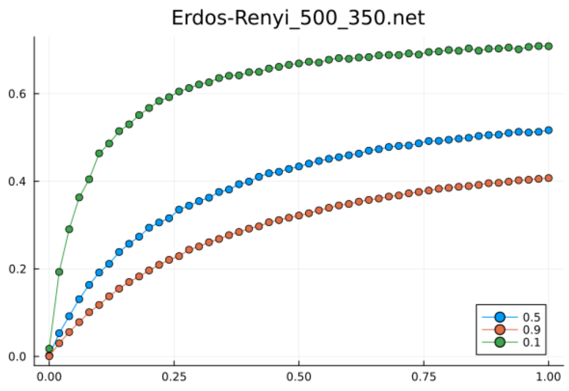
However, this effect is not seen in the next fig 4.2, which uses two generated networks with the Configuration model. Where it seems that both plots are behaving the same way, even when they have different parameters for their creation.



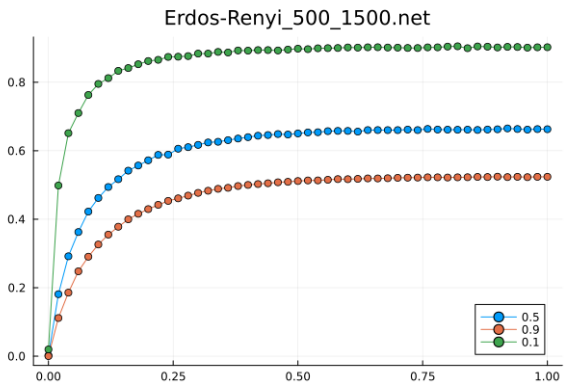
(a) $ER(500, 25)$



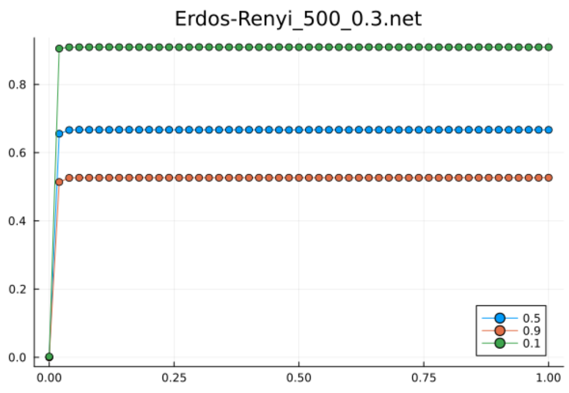
(b) $ER(500, 50)$



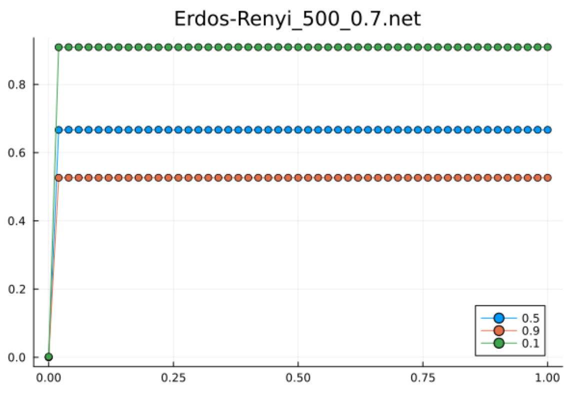
(c) $ER(1000, 350)$



(d) $ER(500, 1500)$



(e) $ER(500, 0.3)$

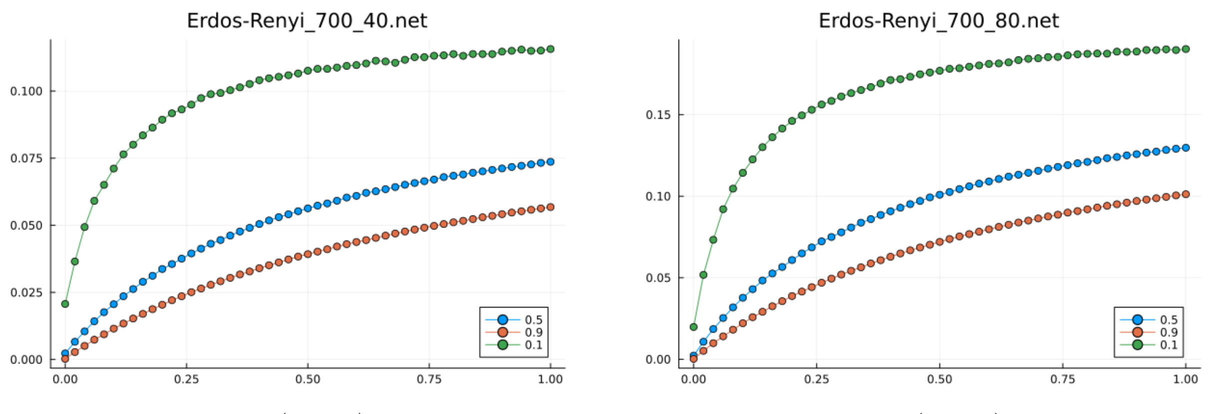


(f) $ER(500, 0.7)$

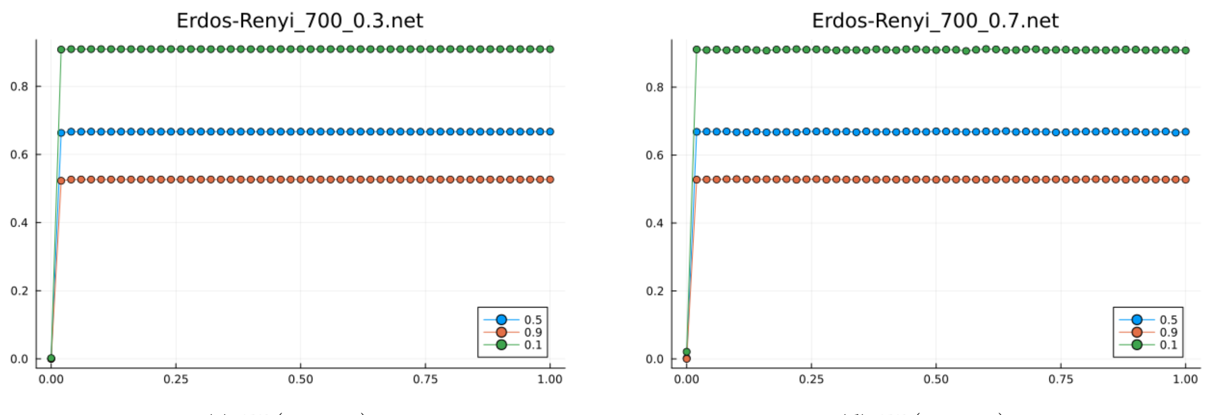
Figure 4.3: Erdos-Renyi Networks of 500 nodes plot ρ vs β at different μ values. Plots are networks generated a, b, c and d using the $ER(n, k)$ version and e and f using the $ER(n, p)$ version.

When we look at the descriptors table, we notice that they are almost the same, with the same 700 nodes, and a very similar average degree, the only difference seems to be the max value for the plot on the left which is 10, vs the max degree for the plot on the right which is 8. However, since the average degree is practically the same, we can observe that this difference in the maximum degree practically does not affect the dynamics of the epidemic spreading.

As we can see with other models, for Erdos-Renyi networks, as shown in Figure 4.3, it is possible to notice that an increase in β leads to a corresponding increase in ρ. This trend underscores the basic epidemiological principle that higher infection probabilities escalate the spread of an infectious disease within a network. Additionally, the influence of μ is also noticeable; networks with higher recovery rates (exhibit lower ρ values at equivalent β levels.



(a) $ER(700, 40)$ (b) $ER(700, 80)$



(c) $ER(700, 0.3)$ (d) $ER(700, 0.7)$

Figure 4.4: Erdos-Renyi Networks of 700 nodes plot ρ vs β at different μ values. Plots are networks generated a and b using the $ER(n, k)$ version and c and d using the $ER(n, p)$ version.

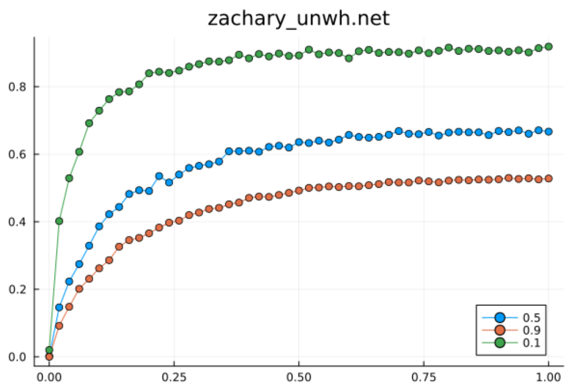
When comparing networks of different densities, it is evident that denser networks (with higher average degrees) show higher ρ values for the same β and μ. For instance, the comparison between

ER(500, 25) and ER(500, 0.7) shows that the denser network reaches higher ρ levels more rapidly as β increases, illustrating how denser connections facilitate a faster and more extensive spread of the infection. This effect is even more pronounced in plots such as ER(1000, 350) versus ER(500, 1500), where despite the larger size of ER(1000, 350), the denser ER(500, 1500) network exhibits a quicker escalation in ρ.

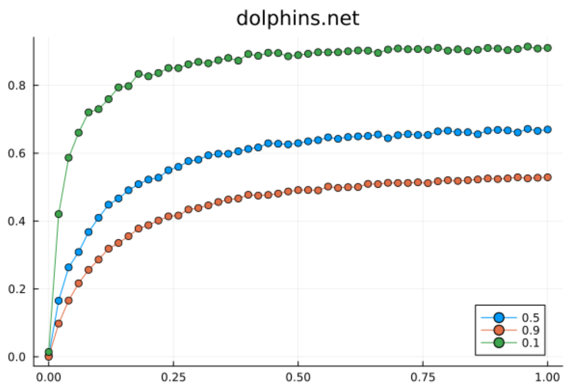
In sparsely connected networks, ρ can remain comparatively low across all β values. This suggests that sparse networks act as a barrier to the widespread transmission of the disease, even when β is high. The role of μ in these kind settings implies that recovery rate becomes a less important factor in controlling disease spread when network connectivity is low.

In the Figure 4.4 the results highlight the role of network connectivity and recovery rates. In denser networks, exemplified by configurations such as ER(700, 0.3) and ER(700, 0.3), there is a marked increase in the fraction of infected nodes (ρ) as the infection probability (β) rises, demonstrating that higher connectivity facilitates more rapid and extensive spread of the disease.

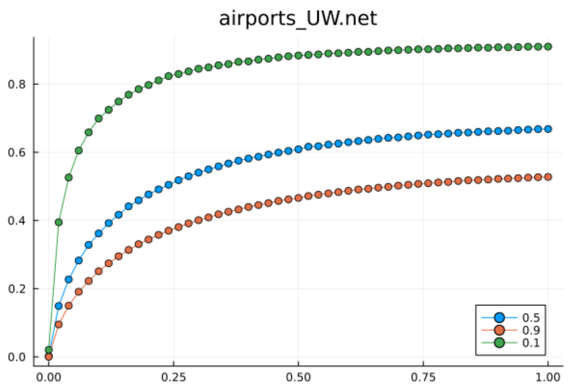
Moreover, the impact of recovery rates (μ) is evident across all network types, with higher recovery rates consistently associated with lower ρ values. This suggests that enhancing recovery rates can effectively mitigate the spread of infections within a network.



(a) Zachary Unwh Network



(b) Dolphins Network



(c) Airports UW Network

Figure 4.5: Erdos-Renyi Networks plot ρ vs β at different μ values.

Finally, when we take a look at the real networks shown in fig 4.5, we can see a clear trend for the Zachary and dolphins networks that are not present in airports: they seem to be 'noisy'. This happens when we see the total of nodes for these networks. The Zachary and Dolphins have very few nodes compared to the airport network.

Apart from that, we can see that they seem to show very similar behavior of the different values of ρ with the increase of β and μ, and this can be explained if we take a look at the table with the descriptors where we see that even when the number of nodes is very different and the maximum degree is also very different, the average degree of all is not. Therefore we can conclude that the average degree is a very relevant parameter on the behavior of the plot.

4 | Conclusions

This assignment provided a comprehensive and deeper understanding of the epidemic spreading, from the different configurations in the SIS model that can represent its dynamics, and how even when it may have some limitations, complex network structures can be very useful for modeling the interactions of the elements of the network. As we saw both, theoretically in class and practically during the implementation of this assignment, this approach can give a good insight into the different behaviors of the parameters, and the actions that can be taken to mitigate the spread.

What may seem obvious at first, the influence of the average network degree on the total number of infected nodes, or the profound impact of the recovery probability, denoted by mu, on infection rates-became concrete and quantifiable through our ability to generate and analyze multiple plots. These visual aids not only reinforced our theoretical knowledge but also highlighted how small changes in network characteristics can lead to significant differences in outcomes.

This exercise helped us to get a better understanding and hands-on experience of complex network dynamics in epidemic spreading demonstrating the applicability of complex network analysis in real-world scenarios. (That nobody, after the global contingency that impacted us just some years ago, could ever doubt about.) The creation of insightful plots also contributed to a deeper and more visual understanding of the behavior of the different parameters. The skills and knowledge gained through this assignment allow us to improve our analytical skills and contribute to our understanding of complex systems.

Also, we highlight the relevance of the knowledge of epidemic spreading and how it can be utilized in far more real-world problems that can be also modeled the same way. It would be very interesting to use this approach with the relevance of social networks today to apply these concepts to other domains.