Complex Networks
Assignment 4: Epidemic spreading on complex networks

Denaldo Lapi and Samy Chouiti

May 1, 2022

Contents

1 Introduction

The goal of this assignment was to apply Monte Carlo simulation of an epidemic spreading dynamics in complex networks, using the SIS model in which each node represents an individual which can be in two possible states:

- **Susceptible (S)**, i.e., healthy but can get infected;
- **Infected (I)**, i.e., has the disease and can spread it to its neighbors.

The main objective of the exercise is the calculation of the fraction of infected nodes, ρ, in the stationary state, as a function of the infection probability of the disease β, for different values of the recovery probability μ.

We'll perform simulations for various types of undirected networks of different sizes and we'll visualize the plots of $\rho(\beta)$

As an additional step, we'll compare the $\rho(\beta)$ result from Monte Carlo simulations with the theoretical prediction provided by the Microscopic Markov Chain Approach (MMCA) model. For what regards this model, we would like to remark that we consider a reactive process, i.e. we assume that each node contacts all of his neighbors at each time step, and we formulate the MMCA for the case without one-step reinfections, i.e. only nodes susceptible at the beginning of the time step can be infected. Therefore, the model equation we considered is [1]:

$$p_i(t+1) = (1 - p_i(t))(1 - q_i(t)) + (1 - \mu)p_i(t)$$

The details of all the steps of the implementation of the 2 methods (with the required parameters) are shown in the provided notebook *MC_simulation.ipynb*. We strongly recommend to analyze the notebook, which properly explains the main procedures we used in order to implement the simulation loop in *Cython*.

2 Software

We implemented the simulation process and the MMCA model by using *Python* by means of a *Jupyter Notebook* that was running on Google Colab, in order to run multiple instance at the same time without using our personal computer's computing power.

Due to the high computational complexity of the simulation function, which requires several nested iterations and array accesses, after several experiments and trials we decided to use the [2] extension of Python.

In particular, *Cython* is a superset of the Python language that additionally supports calling C functions and declaring C types on variables and class attributes. This allows the compiler to generate very efficient C code from Cython code. The C code is generated once and then compiles with all major C/C++ compilers.

In our case, we used *Cython* to define the function performing one full simulation and also the function performing the MMCA iterative procedure: by using statically declared C variables and by compiling these 2 functions, we obtained a great improvement on the performance of our algorithms.

For what regards the parts purely related to the network data structure, we relied on the "igraph" package [3] of Python, which is based on C data structures and it's build upon the C language, therefore allowing high performances.

[1] Gómez, Sergio, et al. "Discrete-time Markov chain approach to contact-based disease spreading in complex networks." EPL (Europhysics Letters) 89.3 (2010): 38009.

[2] Cython documentation

[3] igraph documentation

2.1 Structure of the assignment folder

The assignment folder is organized as follows:

- *nets folder*: Contains the networks used to perform the simulations (some of them were generated by using the already available methods of "networkx", while the others were taken from the networks provided in the previous assignments)
- *MC_simulation.ipynb*: Python notebook with all the functions and steps performing the MC simulation and the MMCA iterative equations. The notebook is run for a single sample network. We strongly suggest to visualize this notebook to get a better understanding of the report.
- CM_model_mod.ipynb: Python notebook used to generate Configuration Model networks (taken from Assignement n°2)
- *Report.pdf*: The present report

3 Computation method and Plots

In this section, we will present the results of the computation of the SIS model using both Monte-Carlo (later refered as MC) simulations and Microscopic Markov Chain Approach (later referred as MMCA). Particularly, the following computations will be comparing those two approaches and allow the study of the evolution of ρ[4] with β including different μ. The different parameters of our simulation are shown below 1.

Name	Value	Meaning
P_0	0.2	Initial infection probability of the network
N_{Rep}	100	Number of simulations
T_{Max}	1000	Number of total timesteps of a simulation
T_{Trans}	900	Number of timesteps of the transitionary state
β	0 to 1 with $\delta\beta = 0.02$	Probability of infection
μ	$\{0.1, 0.5, 0.5\}$	Probability of recovery

Table 1: Parameters used for our simulations

3.1 Computation method

The first step to compute those simulations was to generate the networks that the simulations were going to be applied on. To do so, for networks based on models (CM, ER, BA, etc.), we used the generators that we conceived in previous assignments [5]. The networks were then initialised with an initial infected probability of $P_0 = 0.2$.

When done so, we had to perform simulations on the network [6] using discrete time steps for $T_{max} = 1000$. Because the evolution of the simulation was composed of a transitioning state of time T_{trans}, we saved the ρ value as a mean of all values of the stationary state.

[4] Ratio of infected individuals (vertices) over the whole population (network)

[5] igraph (and networkx) include native generators for all famous models, including ours. However, for CM for example, we added post-processing of the network in *Assignment 2*, with removing self-loops and multi-edges, thus we used our previous work to do so.

[6] Following either MC or MMCA rules, presented above

This simulation was performed $N_{rep} = 100$ times over each network in order to compute the mean value of ρ of the network (as keeping the value for only one simulation may lead to less general insights for a network).

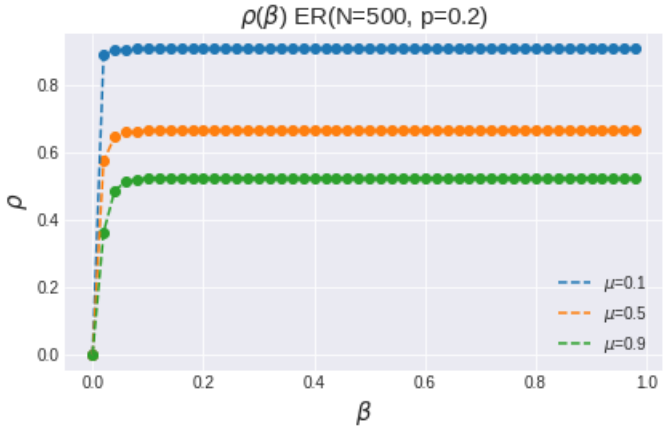
Because we wanted to study the influence of the β and μ parameters, we then iterated the above process over different values of β for a fixed mu: 51 values from 0 to 1 with $\delta\beta = 0.02$.

Finally, the last process was also iterated for each μ values that we fixed to be $[0.1, 0.5, 0.9]$.

Let's now visualize the plots comparing the results obtain through the MC simulation (dashed line) and the MMCA model (sequence of point markers), for the different kind of networks we analyzed.

3.2 Erdos-Renyi generated networks

Compared to other networks, drawing conclusions from the simulation over Erdos-Renyi generated network is pretty straightforward.



We can see that both MC (dotted line) and MMCA (dashed line) are superposed thus making MMCA a relatively accurate model of the simulation performed by MC.

Also, we can see that the infected ratio is inversely correlated to the recovery parameter as we can see that for $\mu = 0.9$, ρ is about 0.5 (for most β values) when having $\mu = 0.1$ leads β to be around 0.85, which is simply explainable by the fact that having higher recovery rates leads to less infected individuals in stationary state. This conclusion will also be drawn in general for all next networks.

Finally, we can also see that $\rho(\beta)$ reach a stationary state starting from $\beta = 0.08$, thus making the infected ratio independent of the infection probability starting from this value.

3.3 Configuration Model generated networks

In overall, simulations ran on CM generated networks (plots report in a tabular format in table 2) leads the MMCA simulations results to be more different to MC than for ER networks for example. Especially when γ tends to be higher, the difference between both simulation is significantly higher: when MMCA seem to provide an accurate model for MC when $\gamma = 2.3$, those simulation are producing too different results on $\gamma = 3.1$ making the model not accurate anymore (which is even worse with $\gamma = 3.5$).

Also, we can see that for low β values and $\mu \in [0.5, 0.9]$, there is a initial state with $\rho(\beta)$ being constant at 0 before going into transitioning state. The latter is continued until $\beta = 1$ (maximum value) thus making $\rho(\beta)$ not reaching any stationary state. However, this is not the case for $\mu = 0.1$ as there is no initial state with $\rho = 0$ as the curve directly enters in stationary state.

The influence of the γ parameter can be also seen with the fact that $\rho(\beta)$ seem to reach the inflexion point faster for lower γ values.

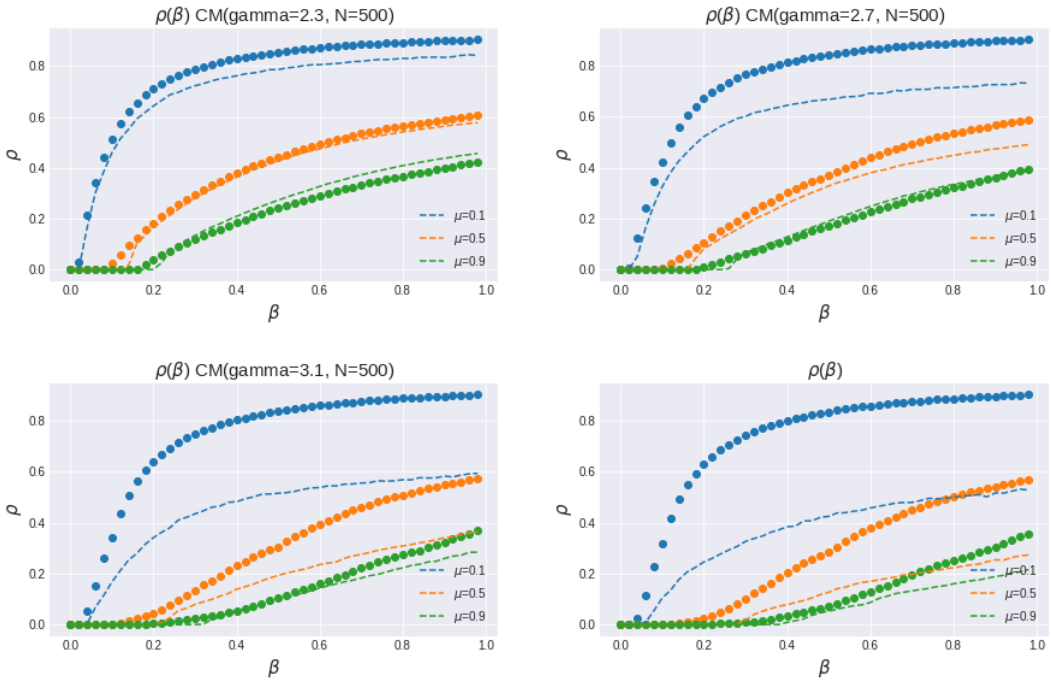


Table 2: Simulations performed over several CM networks (N.B. The last net is a CM with $\gamma = 3.5$ and N=500)

3.4 Barabási–Albert Model generated networks

We performed several MC simulations on BA networks obtained by changing the size of the network (parameter N) and also the m parameter, according to the networkx implementation of the network generator.

The resulting plots are reported as figures inside the table 3.

We can see that the MC dotted line and the MMCA dashed line are very closed to each other thus making MMCA a relatively accurate model of the simulation performed by MC: this happens in both cases, i.e. when changing N (by keeping m fixed to 5) and when changing m (with N fixed to 1000). Again, we can see the same inverse correlation between the infected ratio and the recovery parameter.

We can also notice that for high value of μ (0.9), we have a little offset between the 2 lines, in the area of the plot between $\beta = 0.2$ and $\beta = 0.8$. This offset becomes more relevant in the case of small value of the m parameter, i.e. $m = 3$, where we can also see that the curves grow more smoothly at the beginning (w.r.t. the cases with higher m, i.e. $m = 5$, $m = 7$ and $m = 9$).

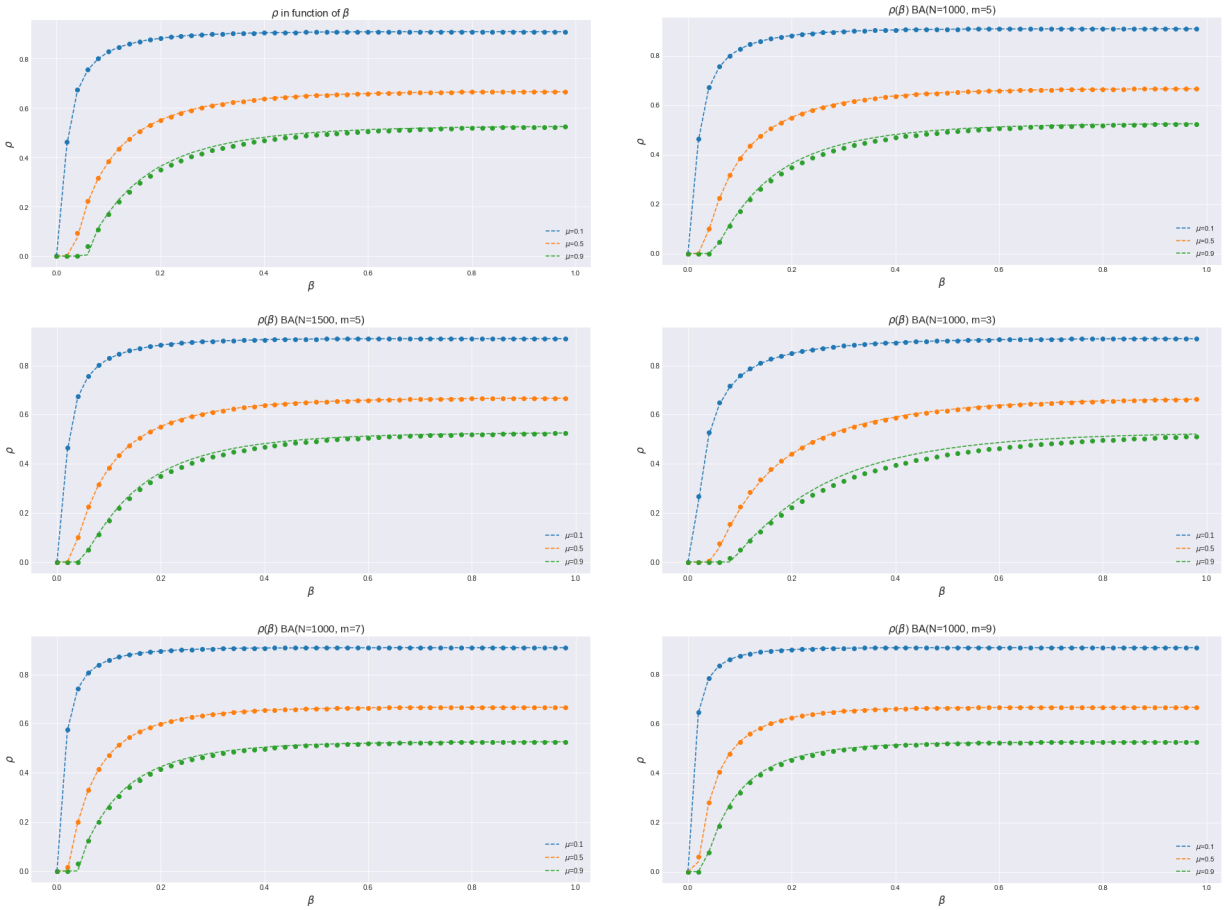


Table 3: Simulations performed over several BA networks (N.B. The first net is a BA with N=500 and m=3)

3.5 Real life example: Airports UW

The first observation that can be made is that the MMCA is relatively well modeling MC, although having an increasing offset for $\mu = 0.9$.

Also, $\mu = 0.1$ seems to be the only one to reach a stationary state with ρ around 0.85 when β reaches its maximum, as other μ values seems to make $\rho(\beta)$ continually evolving.

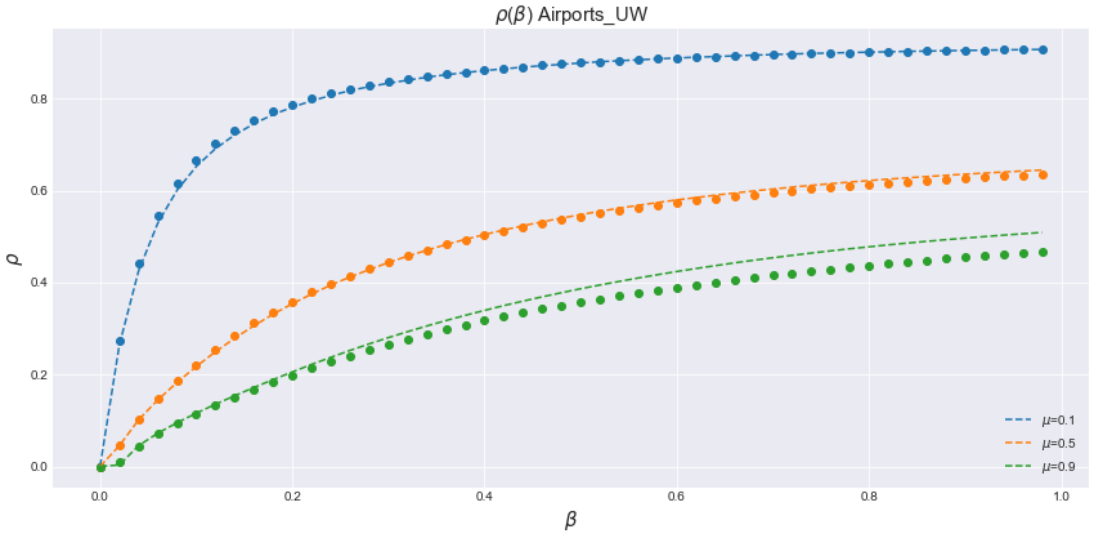


Figure 1: Air transportation network

4 Conclusions

We can see that one general observation can be made over all networks: having a lower recovery rate leads the infection ratio to be higher, which makes sense as less individuals tends to recover from the infection.

Also, to compare MC and MMCA simulation, we can see that some networks were well suited for modeling MC with MMCA (as for Erdos-Renyi for example) when it could also provide less accurate modeling for other models such as Configuration Model.